программирование как бизнес

спец ХАКЕР

спец ХАКЕР

№04(53) АПРЕЛЬ 2005

ЕЖЕМЕСЯЧНЫЙ ТЕМАТИЧЕСКИЙ КОМПЬЮТЕРНЫЙ ЖУРНАЛ

Buck$ Ware

БОНУС

Тест PCI-E видеокарт

Программирование как бизнес

НА CD AlphaControls 4 ■ LMD ElPack 4 XML Parser 2.4.17 ■ FAR 1.70 beta 5 Inno Setup 5.0.8 (Win32/исходники) ■ Nero InfoTool 3.0 Process Explorer 9.01 ■ PuTTY 0.57 ■ FAR 1.70 beta 5ACDSee PowerPack 7.0.62 ■ WinRAR 3.42

(game)land

ISSN 1609-1027

9 771609 102006 04>

CD-CONTENT

- Спец 02(51), *nix без проблем
- Хакер 02(74)
- Железо 02(12)
- Мобильные компьютеры 02(53)
- Обновления для Windows за месяц

Хорошо, когда любимое занятие приносит не только деньги, правда? И если ты решил посвятить себя коммерческому кодингу, то держи в помощь себе диск, содержащий массу полезного софта, который облегчит тебе жизнь и принесет много удовлетворенных клиентов :).

И ЕЩЕ: весь софт из номера!

КОНСТРУКТОРЫ ИНСТАЛЛЕРОВ

- Microsoft Visual Studio Installer 1.1
- NSIS Installer 2.05
- Setup Factory 7.0
- Inno Setup 5.0.8 (Win32/исходники)

ЛЮБИТЕЛЯМ СКИНОВ

- Almediadev BusinessSkinForm (Delphi/Builder)
- Almediadev DynamicSkinForm (Delphi/Builder)
- Almediadev ThemeEngine 5.05 (Delphi/Builder) + набор скинов от Almediadev
- AlphaControls 4
- SUIPack 5.7
- SUISkin 2

STUFF

- .GNU 0.6.12
- Mono 1.0.6
- HelpScribble 7.4.2
- LMD ElPack 4
- Tnt Delphi UNICODE Controls
- XML Editor 1.02 build 101
- XML Parser 2.4.17
- Microsoft XML 4.0 SP2
- expat 1.95.8 (XML parser для PHP)
- Xerces 2.6.0 (на C++/Java для UNIX/Win32)
- SMATCH 0.50 (патч для GCC)
- SPLINT 3.1.1 (UNIX/Win32)
- Help&Manual 3.60
- FAR 1.70 beta 5
- WinRAR 3.42
- PocketRAR 3.41
- TheBat! v3.0.1 Professional
- ZoneAlarm Pro

СОФТ ОТ NONAME

- Nero InfoTool 3.0
- WinTools.net Classic 5.3
- Process Explorer 9.01
- AnVir Task Manager 3.7
- jv16 PowerTools 2005 1.50.271
- BestCrypt 7.12.01
- MenuetOS 0.78 pre6
- Fairstars Audio Converter 1.50
- PuTTY 0.57
- HyperSnap-DX 5.62.05
- DriverCleaner Pro 1.0
- CDCheck 3.1.3.1b
- Semonitor v3.0
- ACDSee PowerPack 7.0.62
- CCleaner 1.17.90

Все это на ЗАГРУЗОЧНОМ CD!

INTRO

Существует множество способов заработка. Можно стать менеджером, а можно ученым, технологом, кондитером, экономистом, юристом, программистом - кем угодно. "Ассортимент" профессий самый широкий. Совет тут может быть один: сделать так, чтобы выбранное занятие приносило не только финансовое, но и моральное удовлетворение. Именно тогда ты сможешь создавать что-то принципиально новое независимо от того, в какой области работаешь.

Создание программного продукта - большое искусство, креатив, созидание! Мы расскажем тебе, как заработать своими собственными мозгами в удивительном мире информационных технологий. Покажи миру свой талант программиста во всей его красе! Помни, что хорошую программу всегда можно продать и заработать на этом немало денег. Но на таком нелегком пути тебя поджидают множество препятствий. Мало просто написать отличную программу – потребуется еще "раскрутить" ее, зарегистрировать на различных ресурсах, защитить от взлома, написать грамотную документацию, создать дистрибутив... Кроме того, важно правильно в юридическом отношении оформить свои права на разработку и решить еще множество проблем.

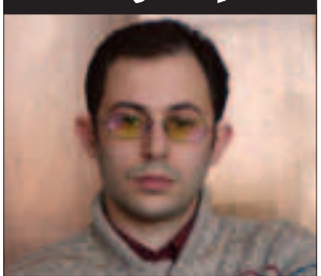
Чтобы помочь тебе в преодолении такой "полосы препятствий", мы подготовили этот номер. Только не забывай, что если ты будешь заинтересован лишь в деньгах, то вряд ли достигнешь больших высот. Кодинг - это в первую очередь творчество. Не забывай об этом и читай твой любимый журнал. Добро пожаловать в мир коммерческого программирования!

Ашот Оганесян

СОДЕРЖАНИЕ № 04 (53)

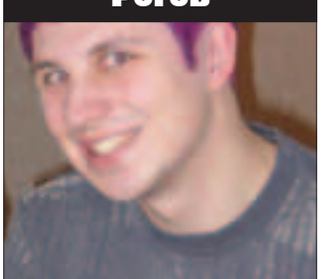
ЭКСПЕРТЫ НОМЕРА

Леонид Кофман



Менеджер по стратегическому развитию компании RealSofts

Евгений "Firstborn" Рогов



Анапитик и специалист по качеству ПО

ЗАРАБОТОК

ТЕХНОЛОГИИ

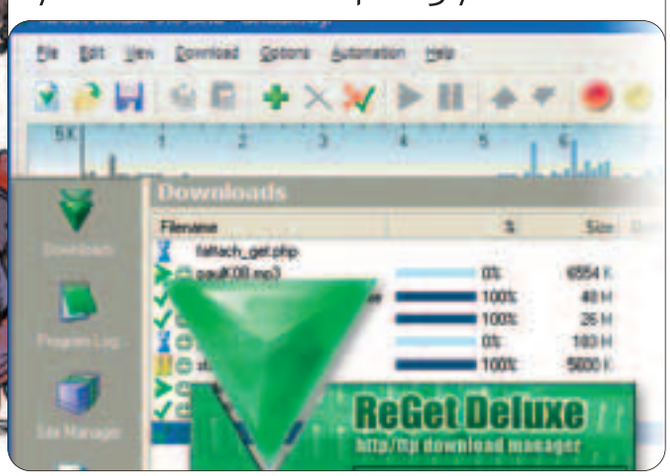
РАБОТА НЕ ВОЛК

SPECail delivery

ЗАРАБОТОК

14 Шароваровареніе: ингредиенты

Как из классного кода сделать коммерчески успешный продукт



ОФФТОПИК

СОФТ

HARD

CREW

STORY

ТЕХНОЛОГИИ

HARD

Редакция
» **главный редактор**
Николай «AvaLANche» Черепанов
(avalanche@real.xakep.ru)
» **выпускающие редакторы**
Ашот Оганесян
(ashot@real.xakep.ru),
Николай «Gorlum» Андреев
(gorlum@real.xakep.ru)
» **редакторы**
Александр «Dr.Klouniz» Лозовский
(alexander@real.xakep.ru),
Андрей Каролик
(andrusha@real.xakep.ru)
» **редактор CD**
Иван «SkyWriter» Касатенко
(sky@real.xakep.ru)
» **литературный редактор**
Валентина Иванова
(valy@real.xakep.ru)

Art
» **арт-директор**
Кирилл Петров «KROt»
(kerel@real.xakep.ru)
Дизайн-студия «100%КПД»
» **верстальщик**
Алексей Алексеев
» **художник**
Константин Комардин

Реклама
» **директор по рекламе ИД (*game*)*land***
Игорь Пискунов (igor@gameland.ru)
» **руководитель отдела рекламы цифровой и игровой группы**
Ольга Басова (olga@gameland.ru)
» **менеджеры отдела**
Виктория Крымова (vika@gameland.ru)
Ольга Емельянцева
(olgaeml@gameland.ru)
» **трафик-менеджер**
Марья Алексеева
(alekseeva@gameland.ru)
тел.: **(095) 935.70.34**
факс: **(095) 924.96.94**

Распространение
» **директор отдела дистрибуции и маркетинга**
Владимир Смирнов
(vladimir@gameland.ru)
» **оптовое распространение**
Андрей Степанов
(andrey@gameland.ru)
» **региональное розничное распространение**
Андрей Наседкин
(nasedkin@gameland.ru)
» **подписка**
Алексей Попов
(popov@gameland.ru)
» **PR-менеджер**
Яна Агарунова
(yana@gameland.ru)
тел.: **(095) 935.70.34**
факс: **(095) 924.96.94**

PUBLISHING
» **издатель**
Сергей Покровский
(pokrovsky@gameland.ru)
» **учредитель**
ООО «Гейм Лэнд»
» **директор**
Дмитрий Агарунов
(dmitri@gameland.ru)
» **финансовый директор**
Борис Скворцов
(boris@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

Web-Site
http://www.xakep.ru

E-mail
spec@real.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.

Content:

ЗАРАБОТОК

Евгений "Firstborn" Рогов (jevgenijsr@gmail.com)

ШАРОВАРЫ НА БОСУ НОГУ

КАК ПЕРЕСТАТЬ ПРОГРАММИРОВАТЬ ТОЛЬКО ДЛЯ ДУШИ

Ты уже освоил любимую среду разработки, имеешь опыт участия в реальных проектах, но вдруг тебя посетило чувство, что это не совсем то, чего тебе хотелось бы... Что делать? Заняться чем-то более интересным или прибыльным? В этой статье я постараюсь помочь тебе найти ответы на эти вопросы.

Какие цели ты ставишь себе? Тебе просто хочется попробовать себя в чем-то новом, тебя воодушевляет реализация собственных идей на практике? Или ты не прочь и подзаработать? Твоя цель, как всегда, определяет все и вся, определяет путь, по которому ты пойдешь. Эта цель определяет, кто ты - любитель или профессионал. Как любитель, ты подойдешь к процессу с вдохновляющей точки зрения: будешь делать такой продукт, который нравится тебе, и сделаешь его таким, каким он понравился бы именно тебе. Как профессионал, ты начнешь с исследования рынка, будешь продвигать такой продукт, который будет пользоваться спросом, и сделаешь его таким, каким его хочет видеть твой покупатель.

Как бы там ни было, твой путь сейчас лежит в страну shareware, или в край условно-бесплатного программного обеспечения, если говорить официально. Почему именно shareware? Потому что это бизнес. Да, это больше бизнес, чем кодинг, хотя и кодингу здесь есть место. А свой бизнес - это свобода как в финансовом отношении, так и в отношении свободы полета мысли: ты сможешь делать то, что хочешь делать сам, а не то, что считает важным дядя-начальник. Заманчиво, правда? Но надо помнить, что до полной финансовой независимости на одном только своем shareware-бизнесе начинающему ой как далеко... Надо когда-то начинать, не так ли? Ну вот и начнем, пожалуй.

Книгу Станислава Жаркова рекомендуется прочитать всем шароварщикам!

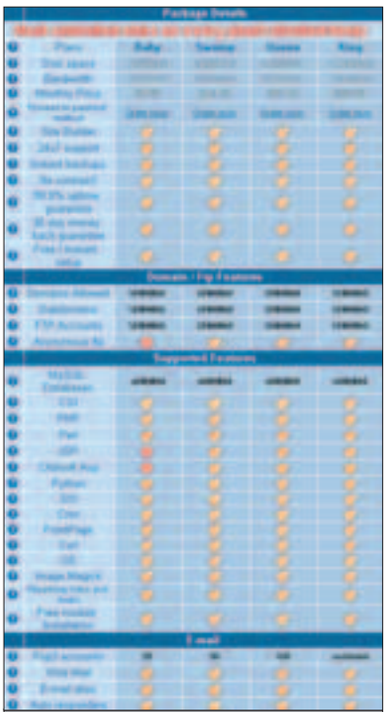
НЕ СОВСЕМ С НУЛЯ

Итак, ты твердо решил встать на путь шароварщика. Тебе предстоит делать множество наверняка новых для тебя вещей, так что перед началом пути неплохо бы удостовериться, что у тебя с собой есть все, что может понадобиться.

Начнем с главного: тебе понадобится много времени, чтобы всерьез развить shareware-проект. Если ты работаешь на двух работах и еще учишься, думаю, не стоит даже начинать. А вот урвать по паре часиков в день после работы - это вполне реально, правда, это будет происходить в ущерб общению с противоположным полом и с алкогольными напитками. Всегда приходится выбирать! Одним словом, время - ресурс невосполнимый, и, начиная строить свою shareware-империю, помни, что само по себе ничего не произойдет, ты должен будешь прилагать ежедневные усилия и тратить на них свое драгоценное время.

Второе, что тебе понадобится, - это деньги. Вспомни, shareware - это бизнес. Он привлекателен тем, что заниматься им может девелопер-одиночка при вполне скромных начальных вложениях, но, тем не менее, они будут нужны. Как минимум, тебе понадобятся сайт и доменное имя второго уровня. Твой бесплатный хостинг не подойдет. Начинать проект без домена и сайта на нормальном хостинге несерьезно, а несерьезность в бизнесе губительна и непростительна. В конце концов, доменное имя можно купить всего за $14,95 (а можно и дешевле - прим. ред.) в год (на http://mapname.com, например), а нормальный хостинг тебе обойдется в сумму $5-10 в месяц. Не так уж страшно, правда? Кстати, раз уж зашла речь о деньгах, позаботься, чтобы у тебя был счет в банке для получе-

Доменное имя второго уровня уже давно стоит вполне приемлемых денег, так что начинать проект без него просто непростительно



Нормальный хостинг вряд ли тебя разорит

ния заветной прибыли и кредитная или дебетная карточка, пригодная для сетевых расчетов оплаты хостинга и домена.

С самым неприятным, кажется, покончили. Теперь составляем небольшой список навыков, которые тебе понадобятся для работы. Итак, ты должен уметь программировать. Язык и среда на данном этапе не так важны, главное – ты должен быть в состоянии писать качественный код с нормальной скоростью. Еще ты должен уметь тестировать. Начинающие программисты часто этим пренебрегают, а зря. В работе над shareware-проектом тебе придется быть и девелопером, и тестером. Причем качество тестирования – далеко не самый последний фактор, в конечном итоге он определяет продаваемость продукта. А еще (о ужас!) тебе придется написать качественную документацию к своему продукту и создать красивый сайт для него.

Внимание: вся писанина должна быть на английском! Увы, тут ничего не поделаешь – ориентироваться на продажи shareware исключительно на просторах необъятной Родины по меньшей мере слишком смело. Если тебя беспокоит финансовая сторона дела, то ориентироваться следует на буржуев с толстыми кошельками и развитую систему платежей. Даже если кто-то из наших соотечественников решит купить твой продукт, он сто раз передумает по вине отсутствия отлаженных систем и средств оплаты (однако если ты ярый патриот, компания RuPay, предоставляющая более 20-ти способов оплаты жителям России и стран СНГ, поможет тебе в получении денег и с твоих соотечественников – прим. peg.). У тебя кредитка уже есть (правда?), а у большинства ее нет, и заводить ее для оплаты регистрации одной понравившейся программки станет мало кто. Нашему человеку, к сожалению, часто будет проще отправится на Google, добавить к названию твоей программы незамысловатое словечко crack и получить все за просто так. Увы.

Вот какие ресурсы тебе понадобятся. Что самое интересное, они взаимозаменяемы! Например, если с начальным капиталом у тебя не густо, зато есть время и нужные навыки, можешь делать все сам и в результате потратиться только на хостинг и домен. А если есть немного деньжат, но с иностранным языком так себе – заплати переводчику, и вот уже у тебя есть и интерфейс, и файл справки на грамотном английском. Если писать тексты для сайта, справки и интерфейса все же возьмешься сам и если к тому же тебе не довелось пару лет прожить в Штатах или в Англии, воспользуйся услугами пруфридеров – носителей языка, которые прочтут тексты твоего сочинения и исправят ошибки и стилистические неточности. Поверь, покупателя очень легко потерять из-за неграмотно написанной первой страницы сайта или грамматической ошибки где-то в интерфейсе.

ОТ ИДЕИ ДО МОДЕЛИ...

Когда-то давно была такая забавная книжка для юных самоделкиных – "От идеи до модели". Причем она тут? Любое дело, будь то выпиливание лобзиком или shreware-проект, нужно начинать с хорошей идеи. И тут возникает вечный вопрос о том, где взять идею? Вопрос сложен, потому что идея она как мед (который, как известно, очень странный предмет): либо есть, либо нет. А откуда приходят идеи, не знает никто. Однако применительно к shareware можно сказать следующее: если некий индивид всерьез решил заняться созданием своего проекта, то, скорее всего, некая идея у него уже имеется. И вопрос в основном заключается в том, как определить, стоящая эта идея или нет? Было бы весьма обидно убить на проект месяц труда, а потом обнаружить, что продаваемость продукта на нуле из-за того, что идея никуда не годилась с самого начала. С созданием программных продуктов так бывает всегда: чем дальше ты продвинешься, прежде чем откроется ошибка, тем дороже она будет стоить. В процессе обдумывания идеи ты можешь понять, что она не так хороша, как тебе казалось сначала, и ты сможешь ее просто отбросить. Если же это обнаружится позже... ну что ж, твой труд, время, а может быть, некоторое количество дензнаков окажутся потраченными впустую. Может быть, не совсем впустую, ты же приобрел бесценный опыт, хотя обычно это мало утешает.

Как проверить твою идею на живучесть и доходность? На самом деле тебе придется провести маркетинговое мини-исследование – выяснить ситуацию на рынке, на который ты собираешься проникнуть. К счастью, в нашем случае это совсем не сложно: тебе в основном понадобятся три вещи: www.google.com, www.download.com и собственные мозги.

Представь, что твоя идея уже реализована. Представил? Представь в подробностях: как будет называться продукт, сколько он будет стоить (ценообразование – это черная магия, но для начала просто возьми какую-нибудь цифру навскидку, надо же с чего-то начинать), как будет выглядеть твой сайт и сама программа, что она будет уметь и как она будет это делать.

А теперь задействуем первые два ресурса из трех вышеперечисленных. Найди своих конкурентов (кстати, их наличие скорее благо, чем зло, но об этом позже), сравни себя с ними. Сколько их? Твой софт дороже, дешевле или стоит примерно столько же? А как с функциональностью? Что нового ты хочешь принести в эту нишу? Чем ты обойдешь конкурентов?

Это что-то должно существовать, иначе просто не имеет смысла начинать проект: уникальная (и при этом нужная!) функциональность, привлекательная цена. Кстати, не спеши сильно снижать цену, потому что, во-первых, регистратор возьмет с тебя комиссию за свои услуги и при небольшой цене на продукт это может быть весьма заметно (например, ShareIt захочет поиметь с каждой продажи или $2,95 плюс 5% от стоимости продукта, или 14,90% от стоимости продукта, но не меньше $2,50). Во-вторых, покупатель, привыкший, что »

Не могу не упомянуть статью Стива Павлины о любителях и профессионалах (название именно такое): www.dexterity.com/articles/shareware-amateurs-vs-shareware-professionals.htm.

465 000 скачек за неделю – есть к чему стремиться!

все продукты в интересующей его нише стоят около $100, может заподозрить неладное увидев, что твой продукт стоит $20 при той же заявленной функциональности. А подозрительный покупатель - не покупатель, он потерянный покупатель.

Очень просто отпугнуть покупателя неправильной ценой. Если она будет неоправданно выше, чем у конкурентов, клиент просто пожалеет денег. Если цена окажется слишком низкой, покупатель решит, что твой софт невысокого качества, что ее производитель несерьезный и вообще стыдно приобретать такую дешевку. Вот ты бы пошел на крутую вечеринку в кедах за пару баксов? Наверное, нет. Как ни странно, у покупателя софта часто срабатывает тот же рефлекс: "Я человек солидный, мне не нужен органайзер за $10, засмеют. Лучше возьму вон тот за $25..." Нередко, кстати, объем продаж возрастает в несколько раз (!) в результате одного увеличения цены с $14,95 до $19,95, в то время как сам продукт никак не изменился! Просто в результате изменения цены поменялся имидж продукта, а это очень многого стоит.

Ты наверняка заметил, что цены на программные продукты (и не только на них) часто выглядят весьма некруглo - $19,95, $24,95 и т.п. Все очень просто: умом понимая, что $29,95 всего на пять центов меньше $30, клиент, тем не менее, подсознательно воспринимает первую цену как гораздо более низкую, потому что $29,95 - это "двадцать с чем-то там", а $30 - "целых тридцать баксов". Такой нехитрый психологический трюк следует использовать всегда, особенно когда он позволяет уменьшить первую цифру цены, как в нашем примере. На выбор цены для твоего продукта будут влиять множество факторов, например, уровень доходов потребителя из целевой аудитории и то, сколько он готов с легкостью заплатить за небольшую shareware-утилитку. Бытует мнение, что американцы воспринимают суммы примерно $10-20 как мелочь, которую можно без особых раздумий потратить на маленький приятный софтик, не отягощенный ничем суперсерьезным, так что многие для начала ориентируются именно на эти суммы.

Когда размышляешь о нишах и о конкурентах, может показаться, что лучший вариант - придумать что-то совершенно новое, то есть создать новую нишу. Там не будет конкурентов, ты будешь сам себе флагман рынка - весьма заманчиво! Не все так просто. Задумайся, почему до сих пор не появилось подобных программ? Может быть, просто потому что они никому не нужны? Залог успеха shareware-проекта в его массовости, ты должен заинтересовать своим продуктом максимально широкие массы. Ты должен быть уверен, что тысячи людей захотят скачать твое произведение. Один из способов вселить в себя такую уверенность - это удостоверится, что подобные программы существуют, скачиваются и продаются. Как нельзя лучше это видно на примере больших архивов программ: посмотри на количества скачек, исследуй структуру каталога, какие бывают категории? А подходящая категория для твоего проекта есть? Если нет, дело плохо, ты попадешь куда-нибудь в Miscellaneous, где твой продукт тихо умрет естественной смертью.

Резюмируя рассуждение об идеях для твоего проекта, остается только констатировать, что все что можно уже придумано и написано, и это хорошо. Парадоксально? Предоставь другим почетное право создавать рынок для тебя, проталкивать абсолютно новые идеи. Как начинающий шароварщик, ты, скорее всего, этого просто не потянешь. Зато ты можешь посмотреть, что делают другие, оценить чужое творчество и сделать лучше! Таким образом ты придешь на подготовленный рынок, будешь уверен в наличии людей, реально заинтересованных в покупке продукта, и вообще будешь чувствовать себя сухо и комфортно.

КОМУ СОФТИНКУ?

Допустим, с идеей ты определился, несмотря на мои отчаянные попытки тебя окончательно и бесповоротно запутать и запугать. Движемся вперед и решаем, подо что и под кого ты будешь писать. Ответ на первый вопрос очевиден, если не забыть о принципе массовости пользователя как об основе успешного shareware-проекта. Угадай с трех раз, какая ОС наиболее популярна в массах? То-то же, Windows. Под нее и будем писать. Я очень уважаю Linux-эксперта в твоем лице, но писать shareware под *nix-системы - занятие неблагодарное. Все дело в том, что тамошнее сообщество, взращенное на фактически бесплатном программном обеспечении с открытым исходным кодом, очень болезненно реагирует на попытки требовать денег за софт. Этот номер не пройдет. Большие компании могут позволить себе эксперименты в этой области, но для одиночки это абсолютно безнадежный вариант. Есть, кончено, и другие операционные системы - тот же Symbian и похожие на него платформы для мобильных устройств. Полагаю, попробовать писать shareware под эти системы при боль-

Решил всерьез разобраться в тонкостях софтварного ценообразования? Начни с "Верблюдов и песочницы" Джоеля Спольски: http://russian.joelonsoftware.com/Articles/CamelsandRubberDuckies.html.

Существует множество регистраторов, тебе придется выбрать одного или двух, так что неплохо бы сравнить предлагаемые сервисы, например, тут: www.blackcat-systems.com/regservices.

А ты уже нашел категорию для своего проекта?

На больших софтварных архивах вроде TuCows представлены разные платформы, но мы будем писать под Windows!

шом желании можно, но весьма рискованно из-за небольшого количества пользователей, несравнимого с размером армии зачарованных Windows'ом. Итак, выбор более или менее очевиден - пишем под Windows.

Кто будет твоим пользователем? В некоторой степени это уже определил выбор платформы: средний компьютерный пользователь-буржуин, использующий компьютер в офисе, дома или и там, и там. Он привык к тому, что за софт надо платить. Он не имеет ничего против этого и готов заплатить и тебе, если твой продукт окажется ему полезен. Наша задача - убедить его, что без твоего софта просто ну никак нельзя жить дальше! Это будет проще сделать тогда, когда ты сам будешь регулярно использовать свою собственную разработку по прямому назначению. Проще говоря, если ты написал систему отслеживания ошибок в программном обеспечении, то во всех своих проектах ты должен использовать именно ее, а не BugZilla, например. Таким образом, ты эффективнее осуществишь тестирование своего продукта, тоньше прочувствуешь его достоинства и недостатки и, в конечном счете, сможешь убедить посетителя сайта своего продукта в том, что из простого любопытствующего посетителя ему неплохо бы перевоплотиться в зарегистрированного! Словом, сама предметная область, в которой ты планируешь развернуться со своим проектом, должна быть тебе не только очень хорошо знакома, но и интересна: только тогда тебе удастся эффективно поддерживать свой энтузиазм на должном уровне для успешного продолжения проекта.

Так что перед тобой стоит непростая задача: сбалансировать свой интерес к разрабатываемому продукту с его будущей продаваемостью. Нельзя бросаться в крайности, нужно попытаться придерживаться золотой середины. И пусть поначалу кажется, что нет такой области, к которой ты мог бы приложить свои силы как shareware-девелопер. Такая область всегда найдется, надо только не бросаться на первую попавшуюся идею, не цепляться за нее, как за соломинку кандидат в утопленники, а вдумчиво анализировать перспективы и, если они печальные, продолжать поиски!

БОРЬБА ЗА ВЫЖИВАНИЕ

А знаешь, кто твой самый хороший помощник в поисках? Никогда не угадаешь - конкуренты! Их наличие есть скорее благо, чем зло, по одной простой причине: конкуренты делают часть твоей работы за тебя. Они готовят для тебя рынок сбыта, они насаждают в головах твоих потенциальных покупателей мысль о том, что такой софт им просто необходим, они невольно подкидывают тебе идейки о том, что стоит делать, а от чего лучше воздержаться. Проще говоря, они совершают ошибки, на которых тебе, человеку безусловно мудрому (раз уж ты решил заняться шароварорерением), можно будет многому научиться. Именно на фоне своих конкурентов ты выделишься своими свежими решениями, более адекватной ценой и мало ли чем еще! А вот не будь их, на фоне кого бы ты выделялся? То-то же.

Давай подумаем, как мы можем использовать злых акул капитализма себе во благо. Во-первых, как индикатор необходимости данного класса программного обеспечения: есть спрос, есть предложение (твои будущие конкуренты). Во-вторых, как генератор идей. Не поленись выкачать все (ну или хотя бы основные) конкурирующие продукты и посмотреть, чем они отличаются друг от друга, чем похожи, чем стараются перетащить на свою сторону пользователя, чего нет ни у кого из них? Может быть, твои соперники начинали в стародавние времена и с тех пор кардинально не меняли пользовательский интерфейс? Тогда, вполне вероятно, тебе будет достаточно продублировать основную функциональность конкурирующих продуктов, сделать все красиво, со скинами, специально заказанными иконками и справочной системой на основе HTML Help. Немного скинуть цену, и вот ты уже заполучил своих покупателей - тех, кому надоели корявый вид других продуктов в этой категории. А может быть, на твой свежий взгляд, всем продуктам в этой категории чего-то не хватает? Восполни этот дефицит, пропиарься на основе этой до недавнего времени отсутствовавшей функции, и вуаля - у тебя опять появились покупатели, которые с радостью поправят твое благосостояние.

Однако не всякий конкурент может быть полезен тебе как начинающему шароварщику. Для примера скажи, ты бы стал писать свою операционную систему и распространять ее как shareware? Ты скажешь, что это самая бредовая идея из всех, что только могут возникнуть в самом воспаленном воображении, и ты будешь полностью прав! А почему? На рынке ОС есть очень неудобные конкуренты: мелкософтовый гигант с его миллиардными доходами и раскрученным продуктом (угадай, каким) и ОС семейства Linux, отличительной особенностью которых является их бесплатность. Шароварщик-одиночка не сможет тягаться с Microsoft и его рекламным бюджетом, даже если напишет ось, которая в сто раз лучше Windows XP. Точно так же он не сможет тягаться с качественным бесплатным продуктом! Кто станет платить, если можно получит то же самое бесплатно? Кроме того, одиночка не напишет ОС за реальное время, что, казалось бы, сводит ценность

Первые четыре позиции в списке Most Popular на Download.com занимают бесплатные программы

этого примера к нулю, но я все-таки надеюсь, что суть ты схватил.

А вот еще несколько подобных примеров. По моему скромному мнению, не стоит начинать свою шароваропи-сательскую деятельность с попыток создать и продать аналог MS Office или Adobe Photoshop просто потому, что эти два вполне коммерческих продукта являются стандартом де-факто, и конкурировать с ними могут только очень дешевые (но функционально не ущербные!) или вообще бесплатные продукты - в нашем случае в роли таковых могут выступать, скажем, Open Office и Gimp. В общем случае наличие в твоей целевой нише известных, функционально богатых и бесплатных продуктов должно настораживать очень сильно. Вряд ли ты сможешь рассчитывать на успех shareware-продукта там, где имеется сильный и к тому же бесплатный конкурент. Не стоит писать медиапроигрыватель, потому что есть WinAmp. Не стоит замахиваться на рынок универсальных просмотрщиков графики, потому что XnView все равно лучше и не стоит ничего. Не пробуй продавать еще один FTP-клиент, потому что есть, например, FileZilla. Хочешь написать и продать ICQ-клиент? Вспомни, что Miranda пока что никуда не делась и исчезать с горизонта не собирается... И таких примеров можно привести море! Я не утверждаю, что тебе никогда не удастся превзойти тот же WinAmp и занять его на данный момент очень даже лидирующее положение. Почему бы и нет? Я просто хочу, чтобы ты осознал, что начинать с этого, наверное, не стоит.

Обобщая все вышесказанное о конкурентах, можно сделать такой вывод: для начинающего шароварщика вроде тебя идеальной будет такая ниша, на которую, во-первых, пока не »

Хочешь пообщаться с живыми шароварщиками? Заходи на www.rsdn.ru, в раздел "Форумы"->"О жизни"-> Shareware, узнаешь много интересного и полезного (особенно если умеешь пользоваться поиском).

Ответы на некоторые вопросы ты наверняка найдешь в русском шароварном FAQ'е: www.shpaga.ru/faq2. Ресурс во многом уникальный, хотя и давно не обновлявшийся.

Разместить свое творение на нескольких популярных архивах - хорошее вложение средств при конкурентной борьбе, особенно если это стоит всего $9 в месяц

обратили серьезного внимания гиганты вроде Microsoft, Adobe, Corel и иже с ними. Во-вторых, в этой нише бесплатных продуктов не должно быть вообще (бесплатные Lite-версии других shareware-продуктов имеют право на существование, поскольку это не самостоятельный продукт, а просто сильно урезанная версия shareware-флагмана). Наконец, в этой нише должны присутствовать как минимум два-три конкурирующих, активно развивающихся и платных (коммерческо-коробочных или shareware) продукта. Так что если ты уже нашел такую нишу и если к тому же она близка тебе по духу и интересам - считай, что тебе повезло!

ERRARE HUMANUM EST

Человеку действительно свойственно ошибаться - древние, как всегда, были правы. Однако ошибаться можно по-разному, и то, как человек реагирует на ошибки, часто определяет успех его предприятия, а shareware'ный бизнес тут далеко не исключение. Я попытаюсь помочь тебе избежать некоторых довольно распространенных ошибок, допускаемых начинающими на этом поприще. Я опять не хочу сказать, что совершив ты одну из таких ошибок - и все, пиши пропало. Все, что тебя не убьет, сделает тебя только сильнее! Однако повторять чужие ошибки совсем не обязательно, лучше ты наделай своих, уникальных ошибок, а от общих я постараюсь тебя если не уберечь, то хотя бы проинформировать об их возможности.

Начнем с основного. Сдается мне, что наиболее частая ошибка кодера, становящегося автором шароварного проекта - это остаться кодером. Будет вполне логично изменив сферу деятельности изменить и свое мировоззрение, так нет же, находятся личности, которые не хотят следовать логике! Они никак не могут понять, что код в шароварном продукте - отнюдь не единственная и даже не центральная составляющая. Да, код очень важен, но ровно настолько же важна и справочная система, и сайт продукта, и интерфейс, и продуманность инсталлятора, и маркетинг! А кодер, не изменивший своего взгляда на вещи, все так же продолжает исступленно писать код и никак не может взять в толк, отчего его продукт не продается при ТАКОЙ функциональности? Кодеру не понять, что пользователь не сможет использовать эту функциональность, потому что справки или нет, или она написана на таком корявом английском, что мама не горюй. Кодер не видит элементарных промахов в маркетинге и не понимает, что о его чудо-программе просто никто не знает, но продолжает заниматься своим любимым делом - кодингом. Это чудесно, но шароварщика из такого типа не выйдет, по крайней мере автора успешного проекта точно. Как лечиться? Отрешиться от кодинга и понять, что твоя shareware - это целый проект, который ты делаешь сам полностью, от и до, вплоть до маркетинга и поддержки, а не только пишешь код. Взгляни на вещи шире. Если ты прирожденный кодер, то тебе, возможно, будет не так интересно заниматься разработкой качественного интерфейса, рисованием иконок или написанием справки, но иногда приходится делать то, что нужно, а не то, что хочется. В крайнем случае можно кому-нибудь заплатить, и он сделает за тебя то, что ты не хочешь или не умеешь делать хорошо. Как бы там ни было, помни, что успешный проект - сбалансированный проект.

Почитывай статьи на www.joelonsoftware.com - это поможет тебе изменить закоренелое кодерское мировоззрение и узнать много нового обо всем, что касается software'ного бизнеса

Вторая типичная ошибка шароварщика - при разработке ориентироваться на свои интересы, а не на интересы потенциального клиента. Это просто непрофессионально. Раз твой проект уже достиг той стадии, когда ты пишешь код, то у тебя уже должно сформироваться понимание того, для кого ты все это делаешь, у тебя перед глазами уже должен быть психологический портрет твоего идеального покупателя, ты должен знать, чего он хочет и что ему понравится. И, самое главное, этим идеальным покупателем не должен быть ты! А часто получается именно так: проект, изначально создававшийся "для себя", постепенно развивается, и на каком-то этапе автор решает перевести его в разряд shareware. Что ж, похвально, только при этом должна изменится и идеология разработки продукта!

Разработчик не должен быть единственным пользователем своего софта. Как же этого избежать? Всегда выслушивай комментарии и пожелания своих реальных пользователей (если первая версия уже выпущена). Если таковых нет, покажи свои наработки друзьям и коллегам, узнай их мнение и обязательно прислушайся к нему. Попробуй найти бета-тестеров, например, пообещай им бесплатную регистрацию за, скажем, три качественных и полных баг-репорта или за предложения по развитию продукта. Выложи бета-версию продукта на любимом программистском форуме, попроси оценить - получишь множество полезных советов от коллег (разумеется, после фильтрации бессмысленных высказываний скептиков и банального флуда), даже если эта версия пока что совсем сыра и вообще больше смахивает на прототип, чем на готовый продукт! Так что не следует стесняться спрашивать мнение других о том, что ты делаешь, а выяснив мнение со стороны, его следует принять как ценную информацию и активно внедрять в практику. Успешный продукт - это продукт, у которого много пользователей, ко мнению которых должен прислушиваться и разработчик.

Третья типичная ошибка: "стать шароварщиком просто, на шароварных продуктах можно легко и быстро сделать большие деньги". Я не стану спорить с тем, что успешный проект может приносить хороший доход. Однако никогда не поверю, что начинающий шароварщик сможет сделать потрясающий продукт прямо с самого первого релиза - так просто не бывает! Как говорится, без труда... В общем, хоть никого ниоткуда вытаскивать и не придется (по крайней мере, в буквальном смысле слова), без серьезного труда не обойтись, особенно если ты планируешь начать этот бизнес с минимальными денежными вложениями. Смирись с тем, что

www.swrus.com - место, где общаются российские разработчики shareware. Есть вопросы? Тебе сюда!

поначалу работать придется много, нужно будет многому учиться (учеба и освоение нового - как известно, одно из самых трудных занятий), а отдача в денежном выражении будет нулевая или мизерная.

Если твоя первая продажа придется на второй месяц после опубликования первой версии продукта, то это очень даже неплохо, даже если эта первая продажа останется единственной в том месяце! Объемы будут расти медленно, даже в том случае, если ты правильно выбрал направление деятельности и создал отличный продукт. А если нет? Никогда не следует надеяться, что занятие shareware-бизнесом принесет тебе несметные богатства немедленно. Принесет, но чуть позже, так что не стоит уходить с работы в надежде прокормить семью с одних только доходов от твоего первого шароварного продукта. Говорят, чтобы спокойно жить с шароварных доходов, автор должен иметь за плечами три-четыре более-менее успешных, активно развивающихся продукта. Впрочем, потребности у всех разные, так что эти цифры весьма условны, как ты понимаешь. Не забывай, что даже серьезные продукты на своих начальных стадиях развиваются весьма медленно в финансовом отношении, даже если разработкой занимается целая компания, потому что рынок программного обеспечения вообще и условно-бесплатного в частности весьма насыщен, насыщается все больше изо дня в день. Поэтому помни, что успешный продукт - это продукт, в который вложено много труда и времени.

ТЕМ НЕ МЕНЕЕ!

У тебя может сложиться впечатление, что не стоит заниматься делом, которое сулит столько трудностей. Что ж, решать тебе самому, однако шароварный бизнес - это то, чем ты можешь заниматься, если как следует захочешь, а также то, что может стать самым интересным предприятием твоей жизни, если ты не бросишь дело на полпути. Часто бывает сложно заставить себя продвигаться вперед, когда так легко отступить. Но если у тебя есть цель, средства обязательно найдутся. Поэтому обдумай все не спеша и приступай! У тебя обязательно получится, в этом я нисколько не сомневаюсь.

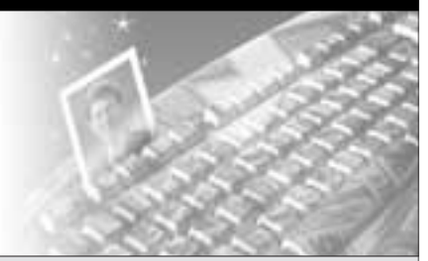
В твоем нелегком труде очень поможет ассоциация шароварщиков-профессионалов - www.asp-shareware.org

Маркетинг всему голова! Если ты не спец в этой области, то найдешь множество интересных идей тут: www.softwaremarketingresource.com.

Если кто-то успешно продает софт, который только и умеет что производить пару-тройку манипуляций с реестром, ты уж точно сможешь стать знатным шароварщиком!

Леонид Кофман (kofman@vlink.ru)

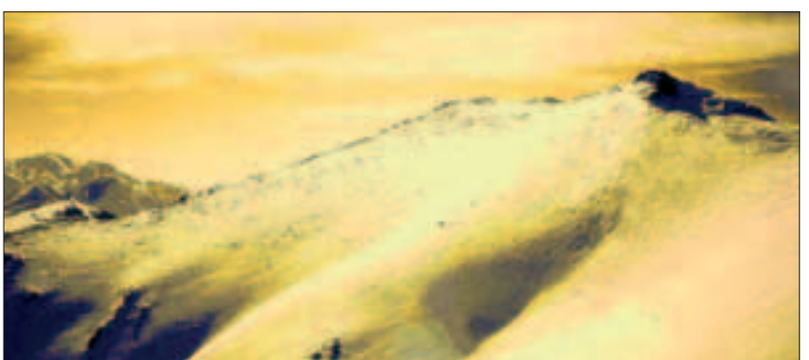
ЗОЛОТЫЕ ГОРЫ SHAREWARE



СКОЛЬКО СЕГОДНЯ ЗАРАБАТЫВАЮТ ЛИДЕРЫ SHAREWARE-ИНДУСТРИИ

Все мы любим считать чужие деньги, это развлекательно и занимательно. А я постараюсь сделать это занятие еще и познавательным. Давай вместе прикинем, какую прибыль может получать программист с продажи shareware.

Сегодня очень многие программисты в России и в странах СНГ занимаются написанием shareware-программ. Достаточно сказать, что в ассоциации SWRUS зарегистрировано почти 2000 программистов (подробнее о SWRUS и других ты можешь узнать в соответствующей статье). Ты можешь смело умножить это число на два, так как отнюдь не все программисты состоят в SWRUS. Я, например, два года спокойно писал и продавал программы и даже не знал о существовании SWRUS. Конечно, не у всех shareware-программистов дела идут отлично. Не все катаются на дорогих иномарках и белых яхтах (кто-то и на синих, вероятно). Но дело не в этом, важен сам факт: любой программист, желающий продавать свои программы, сегодня может это делать. Ему надо лишь сильно хотеть, уметь учиться и не бояться довольно болезненных пинков судьбы. Благо у shareware-программиста есть стимул - неплохой заработок, который мы и попробуем оценить.



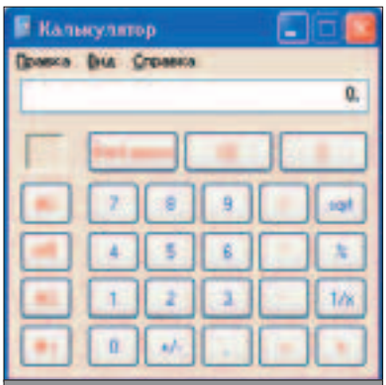
Вот так, наверное, выглядят золотые горы shareware

РАСХОДЫ

Как в любом бизнесе (а разработка shareware - это прежде всего бизнес), прибыль равна разнице между доходами и расходами. Так как твой бизнес начнется именно с расходов, то их обсудим в первую очередь. До тех пор пока у тебя нет денег, самое ценное, что у тебя есть, - это время, причем в отличие от денег, этот ресурс невосполнимый (если только у тебя на антресоли не завалялась машина времени), а значит, время тоже стоит денег. Тут надо все взвесить еще до того как ты начнешь работу в этой области. Ты должен быть готов к катастрофической растрате свого времени.

Для раскрутки программы обязательно потребуется сайт, а это, конечно, тоже расходы, в частности, на web-дизайнера. Если учесть, что сайт должен быть еще и на английском языке, то тебе гарантированы расходы и на переводчика. Когда программа будет готова, прежде чем выставлять ее на обозрение дружелюбной публике, потребуется тестирование, но не самим разработчиком, а посторонними людьми - так сказать, для чистоты эксперимента. Тестерам надо платить (хотя иногда удается получить их работу в обмен на зарегистрированную копию твоей чудо-программы). Ну а потом начнутся самые серьезные расходы: написание пресс-релизов, рассылка их редакторам печатных изданий, размещение программы на сайтах софта (download.com, например, просит за размещение $79). В довершение всего можно заняться оптимизацией сайта под поисковые системы. Расходов много, но они не единовременные. Все сразу делать не надо, мало того, это даже вредно. Как в любом деле, в создании и продвижении shareware всему свое время. Углубляться в каждый из приведенных пунктов расходов я не стану, лучше почитай соответствующую статью. Я, пожалуй, перейду к оборотной стороне нашей неравномерно позолоченной медали - к доходам.

Тебе крайне необходимо научиться ценить свое время



Калькулятор – главное оружие шароварщика :)

Доходы

Сразу предупреждаю: все расчеты, проведенные ниже, не несут в себе цель покопаться в чужом кармане (хотя выглядеть это будет именно так). Моя задача - показать тебе общие объемы и возможности того рынка, на котором ты хочешь работать. Конечно, ты можешь нанять маркетолога, который оценит потенциальный рынок, взвесит возможности конкурентов, даст расчет предполагаемых продаж... и если увидит, что рынок пуст, просто кинет тебя и сам займется разработкой shareware :). Поэтому советую тебе научиться делать подобные расчеты хотя бы приблизительно самому – мало ли что в жизни пригодится, тем более что это не очень-то и сложно.

В shareware-бизнесе есть такое понятие, как коэффициент конверсии. Смысл его предельно прост – это отношение количества людей, скачавших твою чудо-программу, к количеству людей, купивших лицензию на ее использование. Это значение очень сильно колеблется и в первую очередь зависит от того, в какой нише и какого рода программу ты решил продавать. Но об этом чуть позже, нас же интересует, что в среднем разброс коэффициента конверсии составляет 0,1-3%. Конечно, хочется верить, что каждый 33-й пользователь, скачавший твою программу, купит ее (коэффициент конверсии 3%). Но будем пессимистами/реалистами и для наших расчетов используем коэффициент 0,1%, то есть будем считать, что только каждый тысячный пользователь купит программу. Ну и на том спасибо ему огромное. Распространим это наше соображение не только на твою программу (а то как-то обидно получается), но и на предложенные ниже shareware-программы, с помощью калькулятора (еще экстремальней – умножением в столбик) получим основу для наших рассуждений.

По моим наблюдениям, западный рынок shareware закрыт для проникновения постороннего носа, вынюхивающего доходы компаний куда сильнее, чем российский. Достаточно сказать, что ни на одном "их" сайте, продающем shareware-программы, не установлены счетчики, которых в рунете пруд пруди. Иностранные разработчики предпочитают использовать анализаторы логов своего сервера, чтобы следить за посетителями не выдавая им информацию о популярности своего ресурса. Это означает, что тут нам информации не получить.

Зато некоторые данные о популярности сайта разработчика, а значит, косвенно и о его программе, дает поиск в Google в формате link:companysite.com. Но это все же скорее качественный показатель, нам же нужен количественный, а лучше сразу в денежном, причем долларовом, эквиваленте плюс с номерами счетов в банке :). Поэтому куда более перспективным методом оценить чужой доход является подсчет количества скачиваний заданной программы. Количество мы смело умножим на коэффициент конверсии и на цену одной копии, в результате чего получим доход от этой программы за N скачиваний.

Тут сразу хочу оговориться, чтобы не запутать тебя в самом начале. Коэффициент конверсии, который мы приняли за 0,1%, получается только в том случае, если скачавшие твою программу пользователи располагаются в платежеспособных странах. Это, как правило, США и страны Европы, в то время как наши азиатские товарищи (китайцы, вьетнамцы и индийцы) дают совершенно бесполезный процент скачиваний, не приводящий к покупкам, что может очень сильно повлиять на конечную статистику продаж. Кстати, многие разработчики shareware на своих сайтах устанавливают специальные скрипты, которые по IP-адресу фильтруют посетителей, а если они из стран с "неблагоприятным финансовым климатом" (во как сказал!), то просто не дают ничего скачать с сайта (либо не заносят в лог).

Итак, с теорией подсчета доходов и расходов мы разобрались, пора оценить, что же сулит shareware-бизнес помимо болезней пятой точки. На примере нескольких самых раскрученных продуктов shareware-индустрии я постараюсь показать, сколько примерно можно заработать в этой области коммерческого кодинга.

Для раскрутки программы обязательно потребуется сайт, а это, конечно, тоже расходы.

Доходы лидеров

Обойдя ряд архивов софта и записав показатели счетчика скачиваний, я заполучил некоторые данные о заработке лидеров рынка shareware. Но учти, что все данные были получены во время написания этой статьи и до выхода журнала в свет могли измениться. Также не забывай о том, что на сайтах, используемых в этом анализе чужих доходов, подсчет скачиваний ведется по-разному. Например, на download.com статистика ведется с момента размещения у них первой версии программы (при этом в их базе не хранится дата первого размещения, а только дата последнего), в то время как на BetaNews.com при обновлении программы статистика сбрасывается и подсчет начинается заново.

В этом плане BetaNews.com нам интересней, и считать будем именно по нему, так как тут ясно виден период времени, за который ведется подсчет статистики. Однако Download.com я не мог обойти вниманием, потому что это ну ОЧЕНЬ известный и крупный архив, а данные по этому сайту представлены в таблице чисто демонстративного характера, чтобы ты мог оценить размах некоторых проектов.

Смотрим на таблицу и что же мы видим... надо срочно писать файловый менеджер или программу для записи »

Говорят, если не хочешь работать на дядю 40 часов в неделю – работай на себя 80. Эта фраза на 100% точно характеризует рынок shareware.

Download.com – ну ОЧЕНЬ крупный архив shareware-программ

BetaNews.com – очень удобен для сбора первичных сведений о конкурентах

Название программы	Дата начала подсчета / Количество скачиваний с BetaNews.com	Количество скачиваний с Download.com	Цена, $	Доход
WinRar 3.42 **www.win-rar.com**	26 декабря 2004 126,805	22,787,858	29	$3,677 за 2 месяца
Total Commander 6.50 **www.ghisler.com**	18 января 2005 2,435,332	Нет данных	34	$82,801 за 8 дней
ACDSee 7.0.61 **www.acdsystems.com**	9 декабря 2004 26,412	5,506,973	49,99	$1,320 за 48 дней
ReGet Deluxe 4.1.242 **www.deluxe.reget.com**	20 декабря 2004 26,300	745,804	29,95	$787 за 37 дней
The Bat! Professional 3.0 **www.ritlabs.com**	1 декабря 2004 55,372	199,477	45	$2491 за 57 дней
Nero Burning Rom 6.6 **www.ahead.de**	17 января 2005 329,294	Нет данных	69,99	$23047 за 9 дней
Kaspersky Anti-Virus Personal Pro 5.0.19 RC **www.kaspersky.com**	19 января 2005 30,182	831,126	66,5	$2007 за 7 дней
CuteFTP 6.0 **www.globalscape.com**	29 марта 2004 26,962	14,296,757	39,99	$1078 за 10 месяцев

Ориентировочный доход лидеров индустрии

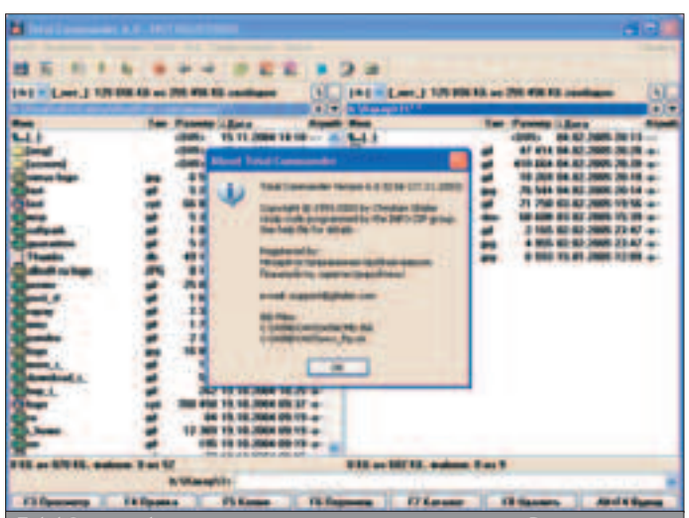
Доход автора Total Commander'а после выпуска очередной версии за 8 дней продаж составил $82,801.

CD :). Еще надо учесть, что это расчеты только по одному пусть и популярному сайту. Если внести сюда еще с десяток файловых архивов средней руки, боюсь, ты не дочитаешь эту статью и побежишь писать свою программу. Однако прежде чем ты ринешься к компьютеру, хочу задержать тебя еще чуть-чуть. Дело в том, что сейчас просто нет смысла писать программы, клонирующие уже существующие популярные аналоги. Вряд ли ты или кто-нибудь другой сможете выдержать конкуренцию с уже завоевавшими славу и почет программами. В этом случае надо искать что-то свое, уникальное и интересное. Притом даже если в твоей программе будет больше всевозможных "фич", чем, скажем, в Total Commander'е, это не обеспечит переход пользователей с TC на твою программу. Если человек привык к программе, ты его за уши от нее не оттащишь (для эксперимента пусть правши попробуют с завтрашнего дня держать ложку или вилку левой рукой вместо правой и пусть узнают, что такое дискомфорт из-за смены привычного).

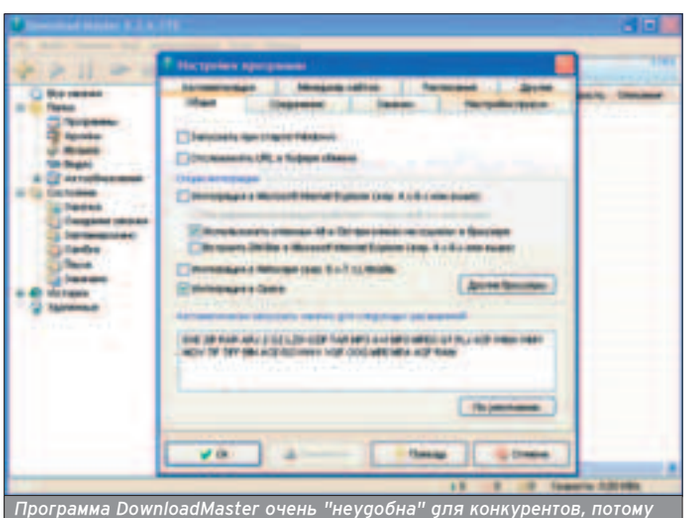
ReGet появился вовремя, потому что подобных ему программ в то время почти не существовало. Сейчас же их очень много, причем стали появляться очень качественные freeware-аналоги, например, DownloadMaster. Ну как тут можно конкурировать?

Кстати, по некоторым данным Владимир Романов (автор ReGet'a) опять начал работать как программист-наемник, и это после стольких лет развития собственного бизнеса! Конечно, автору Total Commande'a Кристиану Гисперу с его доходами это не грозит, однако все равно стоит задуматься. Наверное, не все спокойно в датском королевстве.

Отчаиваться не стоит, хотя бы потому что, в общем, каждому найдется местечко под солнцем. Надо сказать, что тебе придется приложить довольно много усилий, прежде чем твоя программа окупит хотя бы хостинг сайта. Уж на одну-две продажи в месяц может рассчитывать почти любая программа, даже открывалка CD-привода. А вот чтобы продать десяток-другой за месяц, надо уже приложить некоторые умения и старания. Конкретно о том, что это за старания и какие тут умения понадобятся, ты сможешь узнать в других статьях этого номера, от себя же замечу: если ты сам реально будешь пользоваться своей программой, это уже будет первый шаг к ее успеху на рынке, просто потому что ты сможешь яснее представить себе, кому нужна программа, выяснить область ее применения, а следовательно, написать более четкий и мотивированный текст на сайте, более информативные пресс-релизы. Плюс ко всему ты почувствуешь, что твои труды радуют не только тебя. Пожалуй, это самое главное в нашем деле – писать программы, помогающие людям, а деньги придут обязательно: люди готовы платить за качественный софт. Поэтому я уверен, что если за проект взяться серьезно и ответственно, то можно написать хороший продукт, который будет продаваться и приносить своему автору не только чувство собственной значимости, но и реальные хрустящие дензнаки.



Total Commander, судя по всему, популярен не только в России



Программа DownloadMaster очень "неудобна" для конкурентов, потому что она абсолютно ничего не стоит

BUCKSWARE

Евгений "Firstborn" Рогов (jevgenijsr@gmail.com)

ШАРОВАРОВАРЕНИЕ: ИНГРЕДИЕНТЫ

КАК ИЗ КЛАССНОГО КОДА СДЕЛАТЬ КОММЕРЧЕСКИ УСПЕШНЫЙ ПРОДУКТ

Твоя революционная идея нашла свое воплощение в безупречном коде, ты уже рвешься в бой завоевывать просторы шароварных рынков! Похвально, но голый код никто не купит. Почему? Есть тысяча причин, и некоторые из них я постараюсь изложить на нескольких следующих страницах...

НЕ КОДОМ ЕДИНЫМ

Не мне рассказывать, что программный продукт состоит не только из кода. Ты не раз инсталлировал самый разный софт и наверняка вспомнишь, что кроме собственно программного модуля ты кое-что получал в нагрузку. Взять хотя бы сам инсталлятор, который производит установку продукта. Сейчас уже многие известные архивы программ (TuCows, например) серьезно скинут очки твоему детищу, если у него отсутствует инсталлятор и деинсталлятор. Кроме того, как правило, в комплекте с программой присутствует справочная система в том или ином виде. Тебе это может показаться малосущественным: кто будет читать этот help? Пользователи твоего софта, которые, скорее всего, знают о нем намного меньше, чем ты. Вполне вероятно, что они имеют достаточно слабое представление о компьютерах вообще, так что грамотная справка им отнюдь не помешает, а вот ее отсутствие может просто отпугнуть потенциального покупателя, потому что является признаком низкокачественного программного продукта.

И наконец, не следует забывать об интерфейсе пользователя как таковом и о его графическом оформлении - иконках, пиктограммах, кнопках и прочих скинах. Интерфейс - это как раз то, с чем сталкивается пользователь, впервые скачавший твое творение и имеющий намерение с ним сколько-нибудь плотно ознакомиться. А встречают, как известно, по одежке, так что если ты хочешь произвести приятное впечатление своим продуктом, придется слегка попотеть над созданием удобного, прозрачного и красивого пользовательского интерфейса. Итак, поехали!

ТВОЙ ФЕЙС

Для начала несколько слов о том, что твой средний пользователь будет напрямую ассоциировать с твоим проектом как таковым. Речь, как несложно догадаться, пойдет о пользовательском интерфейсе. Как сделать общение пользователя с твоей shareware настолько приятной, чтобы он захотел отстегнуть свои кровные на регистрацию? Задача непростая, но тем не менее разрешимая.

Во-первых, тебе необходимо хорошо представлять себе, кто твой пользователь и чего он хочет от твоей программы. Во-вторых, на основе этого представления ты сможешь составить модель поведения твоего среднего пользователя: какие действия и в какой последовательности он будет совершать. В-третьих, на основе уже имеющейся модели поведения пользователя спроектировать интерфейс, точнее, его первую версию: он должен быть таким, чтобы пользователь тратил как можно меньше времени, а также умственных и физических усилий на выполнение своих основных задач. В общем и целом получается задача минимизации усилий неизвестного индивидуума, который сам не знает, что он хочет от программы и как всего этого добиться от нее же, - что может быть проще? :)

При проектировании интерфейса имеет смысл придерживаться нескольких простых принципов, первый из которых - не делать никаких предположений относительно знаний и умений потенциального пользователя. То есть в общем случае не стоит разрабатывать такую модель взаимодействия пользователя и софта, в которой предполагается, что пользователь - кандидат физико-математических наук. При ориентации на такого пользователя ты автоматически исключаешь из числа своих клиентов тех, у кого квалификация ниже. А зачем тебе это? Твоя цель - добиться как можно большего количества регистраций твоего продукта, чего можно добиться соответствующим увеличением количества пользователей продукта. Принцип массовости - одна из основ удачного shareware-проекта! Так что сделай свой интерфейс таким, чтобы и пятилетний ребенок мог в нем разобраться, тогда и кандидат наук не оплошает.

На практике это означает "графичность" интерфейса. Используй поменьше текста, представь, что пользователь не умеет даже читать! Это, конечно, преувеличение, однако не слишком сильное. Признайся, ты всегда досконально изучаешь многострочные комментарии к различным элементам управления? Думаю, вряд ли - скорее всего ты быстро схватываешь суть и делаешь свой выбор Yes/No или OK/Cancel. А если так, зачем писать много текста? Вполне достаточно короткого, но хорошо продуманного предложения вместо целого абзаца пространных рассуждений. Их, кстати, ты можешь смело выносить в файл справки, соответствующая секция которого будет доступна по нажатию кнопки Help в том окне программы, с которой пытается разобраться несчастный пользователь. А для него, возможно, язык твоей программы просто не является родным и потому сложен для восприятия. Так замени текст графикой! Нарисуй зеленую галочку вместо Yes и красный крестик вместо No, и человек, говорящий на любом языке планеты, быстро поймет, что от него требуется. Разумеется, не все понятия можно заменить однозначно понятной пиктограммой, так что и тут надо постараться не переусерд-

Вот пример неудачного использование текста: его слишком много!

ствовать. Насчет этого порекомендовать что-либо не смогу, разве что призову следовать здравому смыслу.

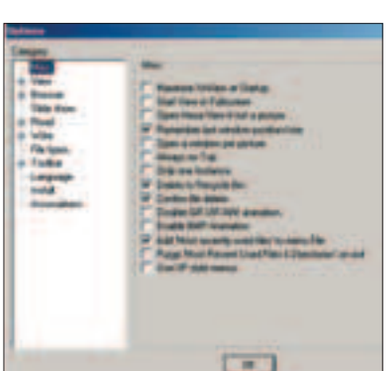
Еще один момент, связанный с применением текста в интерфейсе программного продукта, - это язык, на котором написан текст. Обрати внимание, что если планируется продвинуть продукт в широкие массы, с самого начала его развития надо предусмотреть возможность безболезненного создания локализованных версий. К счастью, для этого существует множество доступных решений (например, для Delphi и C++ Builder - см. www.torry.net/pages.php?id=273), так что технических проблем тут нет (а если и есть, они все подробно освещены в статье о локализации софта - прим. peg.). Однако первые версии твоего продукта вряд ли смогут похвастаться поддержкой нескольких языков, и это нормально. Локализация - дело наживное и требующее финансовых вложений в перевод и в proofreading (исключение: ты полиглот). Ну а первая версия обязательно должна иметь интерфейс на английском языке - это необходимый минимум.

Работая над пользовательским интерфейсом, старайся минимизировать все: не только длину текста, но и количество кликов или нажатий на клавиши для достижения какой-либо цели. В хорошо продуманном интерфейсе оптимизировано даже движение мышки, то есть никогда не получится так, что для выполнения одной из основных и часто используемых функций придется сначала нажать на кнопку в левом нижнем углу окна, а потом тянуться в правый верхний угол. Если модель поведения пользователя проработана на должном уровне, эти две кнопки или пункты меню будут рядом или вообще объединены в одно атомарное действие. Раз оно так востребованно, нет смысла дробить его на части и заставлять твоего драгоценного пользователя делать два клика мышкой там, где вполне достаточно одного!

Кстати, точно так же, как не стоит переоценивать квалификацию и знания среднего пользователя, не стоит думать, что он обладает ангельским терпением и страстным желанием тратить свои силы на освоение твоего продукта. Не надейся, что пользователь будет терпеливо каждый день работы выполнять длительные и монотонные операции в твоем детище. Скорее всего, он просто отправится искать продукт-аналог с более "правильным", то есть подходящим для себя, любимого, интерфейсом. Так что постарайся не утруждать пользователя ничем, не обременяй его необходимостью лишних раздумий. Не перегружай свой продукт опциями, каждая из которых есть выбор для пользователя, требующий умственных усилий, минимизировать которые - твоя основная задача на пути к прозрачному интерфейсу.

Что такое прозрачный пользовательский интерфейс? Тот, который не замечают. Он вроде бы есть, но не требует особых усилий со стороны пользователя, все происходит как бы само собой и, что важно, точно в соответствии с ожиданиями человека по ту сторону экрана. Поэтому постарайся сделать управление твоей программой как можно более стандартным. Нет, я не хочу сказать, что твое творение не должно выделяться из общей серой массы! Скажем так: оно не должно выделяться настолько, чтобы смущать нашего недалекого среднего пользователя. У тебя есть его психологический портрет - сверяйся. Если твой клиент офисный работник, то он, наверное, знаком с Microsoft Office, так пользуйся этим! Организуй свое меню похожим образом, навесь клавиатурные комбинации и иконки а-ля Word на стандартные действия вроде операций с буфером обмена, используй стандартные элементы управления, где только это возможно, и ты увидишь, как твой уже не потенциальный покупатель поразительно быстро разобрался в программе и как он чувствует себя одним целым с ней, а тебе этого как раз и нужно!

Разумеется, есть случаи, когда стандартизация интерфейса не так важна или даже может испортить имидж продукта. Все эти случаи в основном имеют отношение к индустрии развлечений: игры, хранители экрана, мультимедийные проигрыватели и прочие увеселительные примочки. Тут важно "выглядеть" максимально оригинально, свежо. На первый план в этом случае выходит работа художника-оформителя, а не программиста.



Вот так загружать пользователя избыточным количеством опций не надо. Человеческий мозг просто не в состоянии одновременно охватить больше пяти-семи объектов, а тут на одной странице аж 14 checkbox'ов!

Сложно представить себе проигрыватель видеофайлов с обычным прямоугольным окном. Но это не значит, что любой софт должен обладать нестандартным интерфейсом!

И тут на сцену выходят скины - отличная возможность позволить клиенту иметь свою собственную, уникальную и совершенно отличную от стандартной программку, сделанную как будто по его индивидуальному заказу! Натягивание шкурок сейчас принимает масштабы пандемии, и нельзя сказать, что это так уж необоснованно: сам Билли включил поддержку скинов в Windows XP.

Однако у нас здравый взгляд на этот мир, и мы не станем натягивать шкурки на свой проект только потому, что это "модно, йо!". Скины отлично смотрятся на Winamp'e, но попробуй представить, например, корпоративную систему управления конфигурациями под шкуркой в стиле хардкорного хентая - лично мне становится как-то не по себе. Всему свое место, и ориентируясь на серьезного бизнес-пользователя не стоит увлекаться скинами и буйством красок, гораздо выше будет цениться классическая (читай - в стиле Microsoft Office) простота и функциональность. Но если ты взялся написать shareware-игрушку, можешь смело позволить своему таланту художника показать себя во всей красе!

НАРИСОВАТЬ ИЛИ КУПИТЬ?

Однако если с изобразительным талантом дела у тебя обстоят так же, как у меня, то предложение нарисовать графический материал коммерческого качества самостоятельно может прозвучать как издевка. Как же быть? Как всегда в таких случаях, если не можешь сделать сам, покупать! Пусть в Сети и имеются бесплатные коллекции тех же иконок (скажем, на www.kde-look.org - красота!), увы и ах, они бесплатны только для некоммерческого использования, а это не наш вариант. Пусть для стандартного интерфейса несложной программки может хватить иконок, вытянутых редактором ресурсов из того же офиса (благо Microsoft сквозь пальцы смотрит на такие заимствования, если не искажается смысл иконок), однако по край-

Еще раз о дизайне приложений: взгляни на статью Сергея Выдрова (www.rsdn.ru/article/ui/appdesign.xml) или на цикл статей Джоэля Спольски (russian.joelonsoftware.com/uibook/chapters/1.html).

Интерфейс готов, но ты хочешь еще раз убедиться, что он хорош? Отправляйся на Яndex и поищи фразу "usability-тестирование" - вот и список компаний, которые помогут тебе в этом! Не бесплатно, правда.

»

В хорошо продуманном интерфейсе оптимизировано даже движение мышки.

ней мере одна уникальная иконка тебе понадобится - иконка самого приложения! Более того, она должна быть осмысленной и привлекательной, она должна поражать своей лаконичностью, глубиной и утонченностью стиля. И если ты не чувствуешь в себе сил для самостоятельного создания такого шедевра, то тут уж однозначно лучше обратится к профессионалам, потому что в Сети живет огромное множество людей и целых дизайнерских студий, зарабатывающих на жизнь разработкой элементов оформления программ. Найти их несложно: большинство исправно откликнутся на поиск в Google по "icon design" или "custom graphics". Как правило, на сайте такой дизайнерской студии ты сможешь найти примеры работ и оценить их качество, а также определиться с ценой, которую ты готов заплатить. Очень качественную иконку всех возможных размеров и во всех возможных цветовых решениях тебе могут создать всего за $20-35, но иногда можно увидеть и предложения вроде "цены от $1500 за 10 иконок". Так что рекомендовать что-либо конкретное очень трудно - тебе придется самому подыскать дизайнера на свой вкус и кошелек. Радует, что отечественные художники рисуют отличные иконки по вполне разумным ценам, так что совсем необязательно продавать квартиру и машину, чтобы расплатится за дизайн. И потом, может быть, у тебя есть знакомый, который с удовольствием поможет тебе в твоем начинании всего за ящик пива? К слову, немалую помощь в поиске исполнителя графических работ могут оказать онлайновые форумы шароварщиков (вроде того, что существует, например, на www.rsdn.ru), где ты сможешь найти рекомендации конкретных дизайнеров.

На сайте российских шароварщиков (SWRUS) ты найдешь список дизайнеров графики - по крайней мере, будет с чего начать поиск... Тут: www.aklabs.com/swrus/index.php?category=Designers.

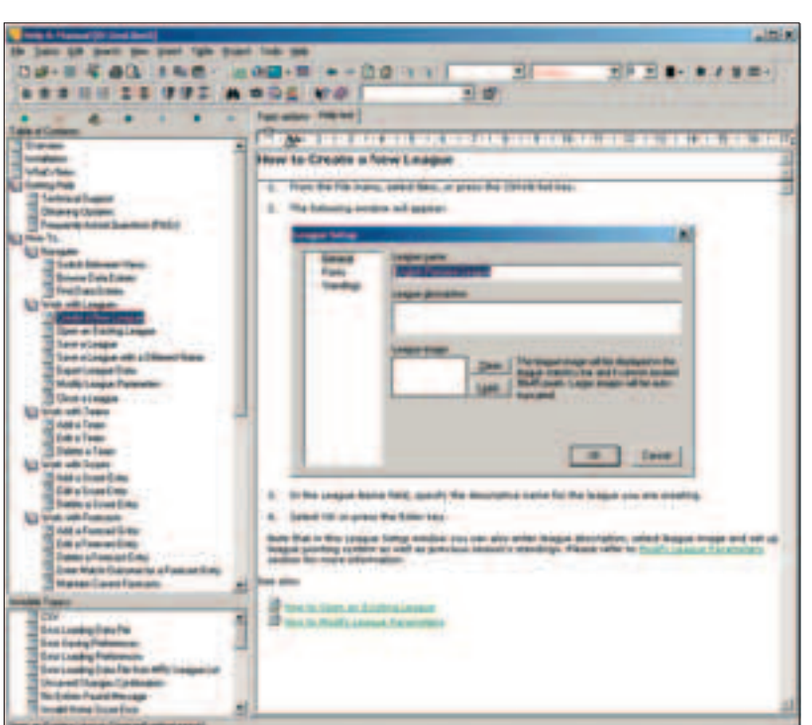
SOS, ОН ЖЕ F1

Твои иконки находятся на стадии разработки, и настало время заняться справочной системой. Пожалуй, глупо отрицать эту необходимость на фоне того, что мы условились считать твоего клиента человеком, мало разбирающимся в чем бы там ни было. Но в то же время не менее глупо думать, что хоть какой-то заметный процент пользователей начнет работу с твоей программой с чтения Help'а (ты сам так хоть раз такое делал?). Так что твой Help почти обязательно должен быть ориентирован на контекстную помощь, то есть в каждом диалоговом окне твоей программы должна присутствовать кнопка Help, нажатие на которую (равно как и судорожное стучание по F1) должно отправлять пользователя в систему помощи, причем не на первую страницу с надписью "Спасибо, что приобрели наш продукт!", а в раздел, имеющий непосредственное отношение к тому окну, из которого была вызвана справка.

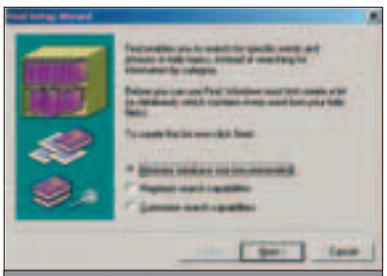
Существует один достаточно популярный метод написания справочной системы такого рода: когда сама программа будет почти готовой, то есть ее основные функции и интерфейс уже будут реализованы, просто пройдись по всем имеющимся в системе окнам и расставь на них кнопки вызова справочной системы. Теперь создавай help-файл, в котором каждой такой кнопке поставь в соответствие раздел, описывающий функцию, реализованную в соответствующем окне приложения. Лучше всего делать это в виде пошаговой инструкции в стиле How to... - так пользователь сможет быстро сориентироваться и понять, на каком шаге он на данный момент находится и что ему делать дальше. Желательно в каждый такой раздел добавить хотя бы по одному скриншоту, благо особые графические дарования тут не нужны, знай себе жми <Alt>+<PrtScr>. Дополнительного графического оформления, как правило, не требуется. Также постарайся ограничить число используемых шрифтов, их цветов и стилей одним или двумя, в противном случае твоя справка будет выглядеть весьма пестро и неудобоваримо. Если ты используешь формат справки, позволяющий создавать гипертекстовые ссылки (WinHelp или HTMLHelp, например), обязательно используй эту возможность: добавляй ссылки на другие разделы там, где пользователь по ходу чтения сможет наткнуть-

Постарайся ограничить число используемых шрифтов, их цветов и стилей одним или двумя, в противном случае твоя справка будет выглядеть весьма пестро и неудобоваримо.

Дизайнерская графика в интерфейсе смотрится очень профессионально, не правда ли?



Подготовить профессиональную справочную систему не сложнее, чем работать в привычном текстовом редакторе



Фокусы WinHelp'a: когда пользователь обращается к нему за помощью, он выкидывает вот такое окошко, где предлагает несчастному пользователю прочесть кучу текста и сделать еще один выбор. Что может быть более раздражающим? К счастью, HTMLHelp избавился от этого недостатка

ся на незнакомое понятие, а также в конце раздела (See also - список разделов, каким-либо образом связанных с данным).

Что касается технической реализации справочной системы, тут нет ничего проще: существуют инструменты, позволяющие создавать файлы справки так же просто, как и документ в Word. Причем один проект можно будет впоследствии откомпилировать как в WinHelp, так и в HTMLHelp и другие форматы - по крайней мере это несложно сделать в моем любимом Help & Manual (www.ec-software.com/hmpage.htm). Ты же для себя наверняка найдешь инструмент по своему вкусу, благо сейчас их развелось множество как бесплатных, так и не совсем. Ну а формат, который ты решишь выбрать для своего файла справки, во многом предопределен: скорее всего, это будет WinHelp или HTMLHelp. Оба эти формата позволяют создавать справочные системы практически любой сложности, разрешают из одного раздела ссылаться на другой, вставлять картинки, использовать различное оформления текста, оба специально разработаны для создания справочных систем - чего же еще? В свою очередь, выбор между этими двумя есть скорее дело вкуса: HTMLHelp новее, моднее, немного шире по возможностям и держит все в одном файле, в то время как справочная система на основе WinHelp может наплодить до пяти файлов за счет полнотекстовых индексов и прочих прибамбасов. Зато, говорят, у старичка WinHelp'a поиск сделан грамотнее, но мне все равно как-то милее HTMLHelp! Хотя бы потому, что он не задает дурацких вопросов при первой попытке что-то поискать в нем.

TO SETUP OR NOT TO SETUP?

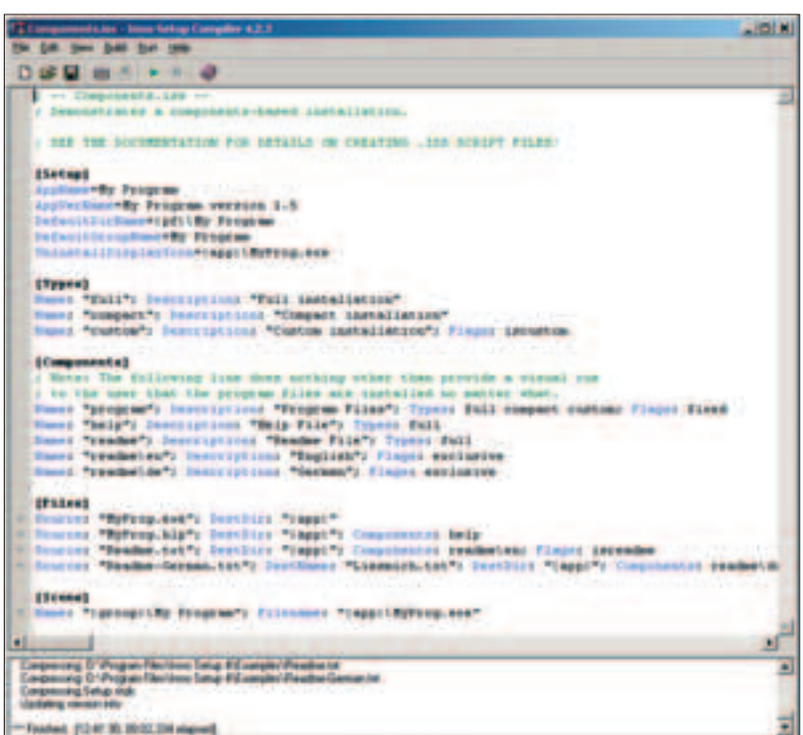
■ Однозначно setup! Повторю, что в shareware-проекте все должно быть прекрасно, от кода до help'a, не является исключением и инсталлятор. Конечно, ты мог бы запихнуть все файлы в один ZIP-архив и выложить его на сайте, но это не будет очень удачным решением, потому что только осложнит жизнь пользователю. Во-первых, у него должен иметься в наличии архиватор. Он есть почти у всех, но тут присутствует ключевое слово - почти. Мы ведь не хотим терять покупателей, правда? Во-вторых, даже если у пользователя есть архиватор, не факт, что он умеет им пользоваться. Тебе это может показаться полным бредом, но таких людей полно! Так что давай не будем спорить о том, нужен ли инсталлятор, а просто сделаем его и сделаем на совесть. А если ты очень хочешь выложить и свой архив с файлами, пусть он будет альтернативным вариантом для скачки для таких оригиналов, как ты :).

На твое счастье существует достаточно большое количество совершенно бесплатных систем инсталляции, бесплатных даже для коммерческого использования. Одним из наиболее распространенных решений в этой области является, конечно же, Inno Setup (www.jrsoftware.org/isinfo.php) - мощный, гибкий и приятный в использовании инструмент, управлять которым ты можешь как путем ручного написания инсталляционного скрипта, так и с помощью гуевого frontend'a, так что с его использованием ты разберешься сам. Не забудь только положить в инсталляционный пакет help-файл и лицензию в текстовом файле (поскольку на юридические документы копирайт не распространяется, можешь подглядеть его содержимое у конкурентов, только не забудь название программы поменять), а также проинструктировать свой инсталлятор создавать линки на все это добро и еще на сайт продукта из Start->Programs->YourSoftware. Вот и все! Хотя нет, вот еще: не называй свой инсталляционный пакет Setup.exe, лучше YourSoftwareSetup.exe безо всякого номера версии в имени файла. Поверь мне, так будет лучше, ты избежишь многих бед.

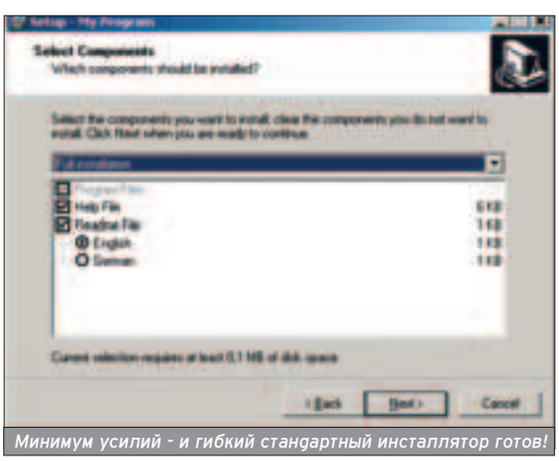
ВДОГОНКУ

■ Должен сказать, что в этой статье мы только проскакали галопом по европам. На самом деле про каждый из затронутых вопросов можно не то что отдельную статью - книгу написать! Как бы там ни было, теперь у тебя должно быть некоторое представление о том, что и как предстоит сделать для того, чтобы твой проект приобрел окончательный товарный вид. На самом деле все совсем не сложно - от тебя требуются лишь желание делать качественный продукт и учиться, учиться и еще раз учиться, как завещал великий "сам-знаешь-кто". А остальное само приложится - не сомневайся!

Если Inno Setup по каким-то причинам тебе не понравился, обрати внимание на аналогичный продукт от производителей Winamp'a, а именно Nullsoft Scriptable Install System (http://nsis.sf.net).



Создать современную систему инсталляции и деинсталляции приложения совсем не сложно, наличие таковой будет воспринято на ура и пользователями, и софтварными архивами!



Минимум усилий - и гибкий стандартный инсталлятор готов!

Каролик Андрей (andrusha@real.xakep.ru)

САЙТ КАК МАРКЕТИНГОВЫЙ ХОД

КАКОЙ САЙТ НУЖЕН ДЛЯ ПРОДВИЖЕНИЯ SHAREWARE

Не секрет, что условно-бесплатные программы продаются преимущественно через интернет. Продажа программы немыслима без ее лица - без сайта. Именно от сайта во многом зависит количество пользователей, которые захотят купить и купят твою программу. При условии, конечно, что сама программа востребована.

Насколько необходим сайт коммерческому проекту по разработке условно-бесплатных программ, особенно если программа всего одна? Некоторые считают, что достаточно ограничиться порталами типа www.download.ru и www.listsoft.ru. И в этом, по мнению многих специалистов, главное заблуждение.

Владимир Тарасов (В. Т.): "Сайт необходим для любого коммерческого продукта. В интернете ты не можешь лично представлять свой продукт, но это под силу твоему сайту. Файловые порталы не смогут донести всю необходимую информацию о твоем продукте, если только, конечно, ты не потратишь уйму денег, чтобы обвешать портал рекламой. И то не факт, что это принесет пользу. Сайт - своего рода визитная карточка для тебя и для твоего продукта. Естественно, это не домашняя страница Васи Иванова - сайт должен выглядеть в соответствии с тематикой продукта".

Леонид Кофман (Л. К.): "Сайт так же важен, как и наличие самой программы. Подавляющее большинство архивов софта требуют прямых ссылок на дистрибутив программы, у себя они файлы не хранят (пожалуй, кроме www.listsoft.ru). Это по крайней мере один из доводов. Сайт еще нужен для того, чтобы получать ценный трафик с поисковиков и чтобы более доходчиво показать пользователю все преимущества определенного продукта, так как на большинстве архивов под описание программы отводится весьма небольшое поле, в котором часто даже нельзя применять html-теги".

Михаил Пеньковский (М. П.): "Если для коммерческого продукта свое представительство в Сети в виде web-сайта является необходимым условием старта, то для условно-бесплатных программ авторов-одиночек сайт является скорее показателем серьезности их намерений относительно собственных разработок. Пользователи элементарно хотят скачать последнюю версию и, наконец, просто узнать побольше о разработчиках. В этой ситуации web-сайт является универсальным инструментом для концентрации информации, доступной в любое время суток. Другое дело, что уровень дизайна сайта и объем информации зависят от возможностей авторов".

Но что должно быть на подобном сайте? Что от такого сайта ждут потенциальные пользователи? Также мы попытались узнать у специалистов, какие ошибки чаще всего допускают по неопытности.

Никита Мелькин (Н. М.): при осуществлении своих проектов придерживается нескольких правил:

❶. Титульная страница сайта рассказывает о задачах, которые помогает решать программа, и о выгодах, которые получает пользователь при ее использования.

❷. Создается отдельная страница с описанием "фич" - конкретных возможностей программы, с описанием технических деталей. Она для тех пользователей, которые заинтересовались программой и хотят узнать детали.

❸. Обязательна страница со скриншотами программы: многие посетители принимают решение только после просмотра скриншотов. Если их нет, просто уходят.

❹. На каждой странице (как правило, в меню) должна быть ссылка на скачивание, которая должна быть хорошо заметной.

❺. Страница регистрации должна содержать максимально подробную информацию о способах покупки.

❻. На сайте должны быть указаны способы связи с разработчиком (или контактный почтовый адрес, или web-форма).

Также Никита Мелькин считает: "Распространенная ошибка - слишком сильный акцент на описании технических возможностей. Пользователю важно решение его проблем, а не ин-

Владимир Тарасов (проект www.ace-clock.com)

Максим Макарский, администратор Кировоградского строительного техникума

Виктор Медведев, разработчик программного обеспечения одной из офшорных компаний Санкт-Петербурга

формация о том, какую замечательную программу ты написал и что она умеет делать. Поэтому описание программы разумнее разбить на две части: выгоды использования и детальное описание технических возможностей".

Максим Макарский (М. М.): "Прежде всего люди ожидают непринужденного интерфейса и быстрой загрузки сайта, а уж потом содержательного описания продукта и его преимуществ. Мой опыт показывает, что намного важнее хорошая начинка сайта (форум, сборник FAQ и т.п.), а не обилие картинок. Хороший пример - сайт компании Mustek (www.mustek.ru - прим. ред.)".

Л. К.: "На сайте разработчика должны быть выложены материалы скрыторекламного характера, мотивирующие человека на покупку. Должна быть страница download, где всегда будет доступна для скачивания последняя версия программы. Страница покупки, на которой доходчиво объясняется, что если человек оплачивает покупку кредитной картой, то ее завтра не засветят на каждом углу, а также о том, что пользователю предоставляется гарантия возврата денег, скажем, в 30-дневный срок".

Виктор Медведев (В. М.): "Ошибка часто заключается в том, что человек хочет продать свою программу и кроме денег его больше ничего не интересует. А посетителю это может не понравиться. Посетитель (потенциальный покупатель) хотел бы знать, с кем он имеет дело".

М. П.: "Не стоит путать сайты коммерческих продуктов и сайты условно-бесплатных программ "авторов-одиночек". В случае с коммерческим продуктом требования достаточно жесткие: полная информация о продукте и его возможностях, секция службы поддержки, онлайн-магазин, информация о самой компании. Это тот самый минимум. В случае с условно-бесплатными программами структура сайта похожа, за исключением онлайн-магазина и службы поддержки, которую может заменить документ, в котором есть ответы на часто задаваемые вопросы. Разница также в объеме публикуемой информации. Основные проблемы и тех, и других - достаточно низкий уровень "юзабилити", то есть удобства в навигации web-сайтов. Порой пользователю для того, чтобы скачать дистрибутив программы, нужно пробраться через несколько страниц, при этом тратить драгоценное время на поиски нужных ссылок. Этим грешат порой и сайты известных компаний".

Михаил Пеньковский, коммерческий директор компании "Агнитум"

В. Т.: "Важно сосредоточится на потребностях пользователя. Представь, что на твой сайт зашел новичок. Что он должен узнать о продукте? Как он должен узнать? К какому выводу/решению должен прийти пользователь-покупатель? Прежде чем начать разрабатывать сайт, стоит ответить на эти вопросы себе самому".

М. П.: "Стоит обращать внимание на единообразие, особенно это касается коммерческих продуктов. Один из главных постулатов брендинга гласит, что должна быть унификация всех визуальных воплощений продукта. Это касается не только web-сайта, но и логотипов, дизайна коробок, визиток, фирменных бланков и т.д. Стоит помнить, что сайт - это один из инструментов, цель которого заключается в предоставлении быстрого и удобного доступа к информации, а также в предоставлении возможности загрузить и купить твой продукт. Некоторые же считают, что цель - выиграть конкурс на лучший web-дизайн и креативный подход. В итоге получаются сайты, от которых нет никакой пользы. Легкость в навигации и доступе к информации, единообразие дизайна являются важнейшими требованиями, которые позволят успешно конвертировать посетителей твоего сайта в покупателей".

Следующий вопрос, которым задаются после разработки сайта, - что необходимо сделать помимо разработки сайта? Как продвинуть уже готовый сайт и вместе с ним - свой программный продукт?

Н. М.: "Особых нюансов нет, рекомендации - это рассылка новостей, пресс-релизов, поиск мест в Сети, где общались бы потенциальные потребители твоей программы, и общение с ними. Из специфических деталей - неплохо сделать массовый "сабмит" сайта по софт-директориям. Если это и не принесет повышения количества посетителей сайта, то может значительно повысить "видимость" сайта в поисковых системах".

М. М.: "Для продвижения программы сначала необходимо добавить ее в максимальное количество каталогов и поисковиков, разместить свою рекламу на форумах, специфика которых близка к предназначению твоей программы".

В. Т.: "Про "сабмиты" и баннерную рекламу, пожалуй, знают все. Также существуют кросс-линки (перекрестные ссылки с других сайтов на твой и наоборот). Здесь все зависит от твоих способностей не только писать код, но и договариваться с другими участниками "рынка". Имеет смысл "линковаться" с сайтами схожей тематики. Важная составляющая на любом этапе разработки - АНАЛИЗ. Анализируй наиболее важные показатели: число скачиваний, число посещений сайта, число покупок, распределение пользователей по различным сегментам и т.п. Делай ВЫВОДЫ, вноси коррективы и в продукте, и на сайте. Нельзя сразу объять необъятное, но и нельзя забывать о таких процессах".

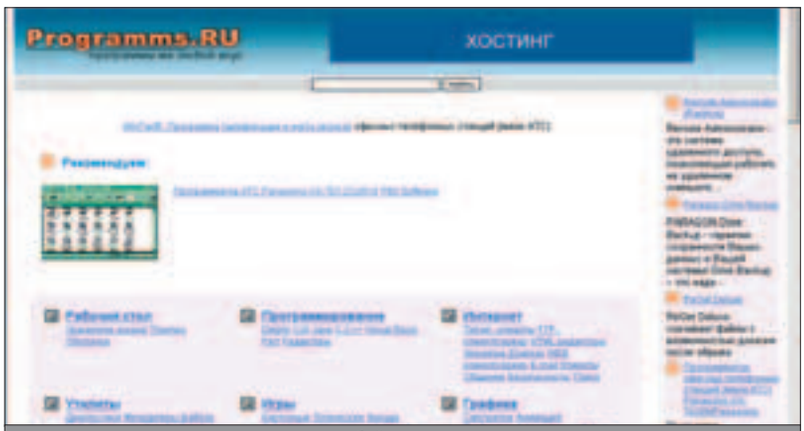
В. М.: "Для продвижения любого продукта я обычно пользуюсь услугами Госкомстата. Там продаются (раньше продавались) книжечки "Промышленные предприятия" какого-либо региона. В них можно найти много интересного. Например, электронные адреса, сайты предприятий, сведения об износе их основных фондов, численность работников, денежный оборот предприятия. Можно сделать вполне адресную рассылку рекламы своей программы".

М. П.: "Стоит продумать, каким образом потенциальные покупатели будут узнавать о продукте. Какими источниками информации они предпочитают пользоваться? Это могут быть поисковые системы вроде Google или Яndex, онлайновые каталоги программ, тематические контент-ресурсы, форумы, службы рассылки и т.д. Не стоит забывать и о традиционных офлайновых каналах, таких как телевидение, радио, журналы и газеты. После определения каналов стоит прикинуть свои возможности размещения информации (не обязательно рекламного характера) в наиболее "отзывчивых" источниках. По поводу PR стоит пояснить. Для автора-одиночки PR-кампанией может стать налаживание связей с журналистами, пишущими по данной тематике, обмен новостями, инициализация обзоров в прессе и консультации по вопросам использования. Не стоит забывать о подходе "партизанская война", когда на выделенных форумах и web-конференциях специально выбрасывается информация и завязывается дискуссия по программному продукту. Это может быть достаточно эффективным инструментом на начальном этапе".

Если твоя программа уникальна, то вышеприведенных комментариев специалистов будет вполне достаточно. Если же у тебя полно конкурентов, то помимо этого необходимо позаботиться о том, как выделить программный продукт среди других и как сделать его более привлекательным.

Н. М.: "Чтобы успешно выдерживать конкуренцию, необходимо выполнение двух условий: программа должна быть качественной и она должна иметь некоторые уникальные возможности. Причем если программа уступает конкурентам по числу возможностей - это не особо критично, но по от-

»



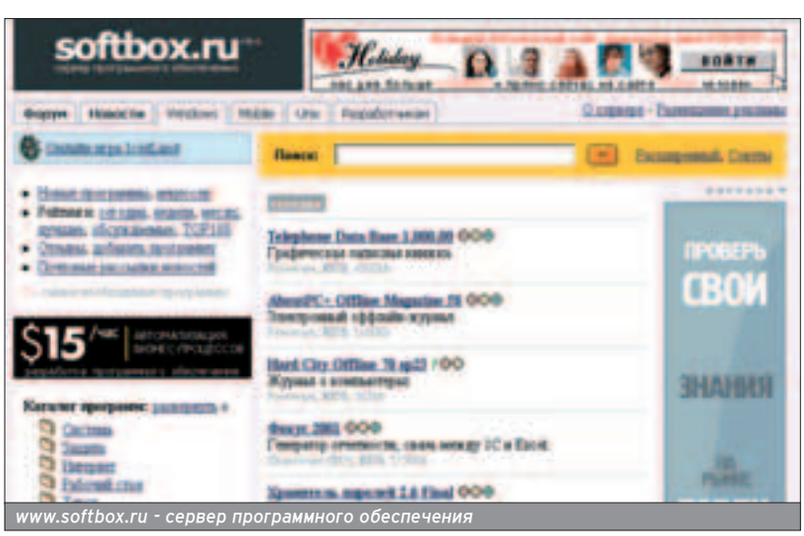
www.programms.ru - программы на любой вкус

soft.mail.ru - каталог софта от известного почтовика

дельным моментам она должна быть лучше/удобнее, что и станет твоим конкурентным преимуществом".

В. Т.: "Знаешь основные принципы/правила в бизнесе о том, как привлечь внимание к себе? Нужно ВЫДЕЛИТЬ и продукт, и производителя. Обязательно! Представь, что на рынке есть уже 100 аналогичных серых продуктов. И появится твой - 101-й, тоже серый. Очень радужные перспективы... А если это будет "желтый" или "красный"? Эффект будет другим. Способы выделения могут быть разными, но важно, чтобы реальные пользователи ощутили, что эти ВЫДЕЛЕНИЯ имеют действительные основания под собой".

М. П.: "Безусловно, нужно выделяться, без этого разработку могут просто не заметить. При этом стоит помнить, что программные продукты - достаточно сложный товар, выбор которого порой обусловлен не только его потребительскими характеристиками (то есть техническими свойствами), но и известностью торговой марки или бренда компании-производителя. Поэтому делая разработки, незначительно отличающиеся от существующих на рынке, рассчитывать на успех рано. Попытайся найти нишу, которая еще не занята ведущими производителями и в которой представленные программы не полностью решают "проблемы" и ожидания пользователей. Torga



www.softbox.ru - сервер программного обеспечения

твоя разработка сможет "попасть точно в цель".

Другой момент относится не к поиску изъянов продуктов-конкурентов и к их учету в своей разработке, как предлагается выше, а к некоторому маркетинговому "трюку". Есть такой термин "латеральный сдвиг", предложенный известным маркетологом Филиппом Котлером. Основная идея - найти уникальное сочетание двух разных понятий. Например, история появления персональных файрволов (брандмауэров), которые стали доступны обычным пользователям. Несколько лет назад этот тип продуктов был предназначен только для системных администраторов и был очень сложным в настройке. То есть бытовало такое мнение, что "файрволы - только для системным администраторов". Осуществляем латеральный сдвиг путем спорной формулировки: "Файрволы - для домашних пользователей". На первый взгляд, в то время это казалось сомнительным. Как можно было сделать файрволы для новичков? Прежде всего упростить интерфейс, скрыть всю техническую начинку внутрь и ликвидировать лишние для домашнего компьютера "механизмы" программы. Налицо инновационный подход, который и был использован компанией "Агнитум" при разработке персонального файрвола Outpost Pro. Такой сдвиг можно попытаться применить к уже известным технологиям, например, для мессенджеров - ICQ, Miranda и др. Широко распространено мнение о том, что "из-за мессенджеров сотрудники тратят время на пустые разговоры на рабочем месте". Делаем латеральный сдвиг: "Мессенджеры не отнимают время сотрудников на пустые разговоры". Как это можно реализовать? Вариантов несколько. Например, позаимствовать у персональных файрволов систему правил, разрешающих/запрещающих разный тип активности. В итоге получаем разработку, с помощью которой можно регулировать количество контактов сотрудников в офисной сети и за ее пределами, а также гибко настраивать эти правила. Для этого мы не изобретаем велосипед, а используем сочетание двух технологий - мессенджера и файрвола. Это первое, что пришло мне в голову. Уверен, что у тебя получится не хуже".

Понятно, что сайт делается не ради галочки, а для достижения определенного эффекта. Степень проявления такого эффекта нужно как-то оценить, чтобы выяснить, достигнута ли поставленная цель. И при необходимости внести коррективы.

М. М.: "Анализировать эффективность действий можно при помощи установленных счетчиков на сайте и по количеству скачиваний программы. Если каждый день сайт посещают, скачивают на нем программу и покупают

■ Леонид Кофман, менеджер по стратегическому развитию компании RealSofts

Второе, что ты должен сделать после того как твоя программа начнет "дышать", - это сверстать для нее сайт. Обязательно помнить, что сайт - это не просто набор страничек, а важнейший маркетинговый инструмент, способный произвести на человека почти такое же впечатление, как и хорошая работа самой программы. Причем с той принципиальной разницей, что сайт всегда можно быстро изменить и привлечь внимание его посетителей к тем или иным преимуществам программы, а недостатки отвести на второй план. Это все в твоей власти и ты обязан этим воспользоваться.

Во-первых, тебе все-таки придется подучить английский. Ну никуда без него, такова жизнь. А все самые важные страницы сайта все равно должны быть отданы на пруфридинг для шлифовки носителем языка: самостоятельно ты не сможешь добиться того же эффекта.

Во-вторых, содержимое сайта должно постоянно обновляться, то есть хотя бы раз в месяц нужно будет добавлять по одной страничке (например, можно добавить интересную тематическую статью или обзор из журнала) и шлифовать текст на остальных страницах. Это крайне важно для привлечения поисковиков, которые более уважительно относятся к изменяющимся ресурсам.

Также на сайте необходимо предусмотреть максимальное количество средств обратной связи с разработчиком. Обязательно нужна подписка на рассылку, очень желателен форум, конечно же, нужна страница саппорта, откуда человек сможет быстро написать тебе письмо.

На страницу покупки обязательно повесить значок с надписью "30 days money back guarantee", для тебя это не будет стоить ничего, так как если человек решит вернуть свои деньги, он их и так вернет через чарджбэк, зато остальных это может привлечь к покупке товара, раз они ничем не рискуют. Необходимо следить за крупными праздниками хотя бы в США и в это время делать скидки и различные подарки: во-первых, потребителям понравишься, а во-вторых, получишь шанс серьезно повысить объемы продаж.

Для продвижения программы необходимо продвигать и сайт, поэтому полезно заняться поиском сайтов схожей тематики и постараться обменяться с ними ссылками. На сайте можно организовать страничку о наградах программы (благо после размещения на архивах софта их у тебя появится достаточно). Правда, сегодняшнего пользователя это вряд ли сильно впечатляет, но может сработать и обратный эффект: если наград нет, значит программа совсем никуда не годится. Не стоит бояться экспериментировать, в конце концов, пользователи тебя не съедят, зато ты получишь бесценный маркетинговый опыт, который впоследствии ой как пригодится.

ее, значит, сайт работает по своему прямому предназначению".

Л. К.: "Сайт - это не цель. Цель - раскрутка программы. И говорить о достижении цели можно тогда, когда продажи программы выйдут на приемлемый для разработчика уровень. Сайт в этом случае является только инструментом".

М. П.: "Если основная цель - пустить в будущем все на самотек, то стоит всерьез задуматься, стоит ли этим заниматься? Если для тебя это хобби, то какое-то минимальное время все же придется тратить, так как хобби не может быть в тягость, верно? Если же это источник дохода, на который ты в будущем собираешься сделать ставку, то начинать это дело с мыслью о будущей "остановке" не стоит. Анализировать же эффективность действий стоит с точки зрения соотношения затрат и доходов, а также соответствия достижений намеченным целям. В случае с условно-бесплатными программами можно анализировать темпы увеличения количества загрузок твоей программы, а также рост количества упоминаний о ней в разных источниках, получение наград в онлайн- и офлайн-прессе и т.п.".

И конечно, обращай внимание на чужие удачные проекты, собирай рецепты успеха, советы и рекомендации. Порой не требуется изобретать велосипед, достаточно внимательно посмотреть на то, что уже есть: над другими проектами сидели и думали умные люди, они тратили собственное время и силы. Анализ чужих ошибок - еще более полезное дело. Копировать находки могут практически все, а делать из недостатков преимущества - единицы. Точнее, некоторые даже об этом не задумываются.

Н. М.: "Среди прочих факторов, влияющих на раскрутку, я бы отдельно выделил техподдержку. Если пользователь быстро получит ответы на свои вопросы, то вероятность превратить его в покупателя возрастает в несколько раз. К примеру, на вопросы о своих программах я стараюсь отвечать максимально оперативно - от нескольких минут до нескольких часов, но не больше суток. Пользователи ценят это, причем выражают свою благодарность в виде многих рублей и не только рублей".

Л. К.: "При продвижении сайта не стоит заниматься массовой рассылкой на FFA-сайты, которые размещают у себя любые ссылки, но со временем удаляют их из базы, заменяя новыми. У меня был подобный печальный опыт, за который я поплатился баном Google (исключение из каталога www.google.ru - прим. ред.)".

М. П.: "Во-первых, надо всегда стремиться к лучшему результату. Позитивный настрой - одно из главных условий для хорошего старта. Во-вторых, ставь перед собой четкие цели и планируй, как ты будешь добиваться их. Не забывай рассматривать каждый вариант решения под разными углами, критикуй сам себя, чтобы найти идеальное решение. В-третьих, помни, что успешный маркетинг и продажи программного продукта не сильно отличаются от распространения других товаров. Если учитывать все это, без знаний маркетинга и технологий продаж тебе будет тяжело добиться серьезного успеха. Поэтому имеет смысл изучить азы маркетинга и почитать литературу, посвященную технике прямых продаж. И в-четвертых, несмотря на все вышесказанное, в любом бизнесе не обойтись без удачи, которой я тебе и желаю!"

BUCKSWARE

Степан Ильин aka Step (step@real.xakep.ru)

КРУЧУ-ВЕРЧУ

КАК РАСКРУТИТЬ И ПРОДАТЬ СВОЮ ПРОГРАММУ

Для хорошего программиста не составляет труда написать толковую программу. Куда больше трудностей и проблем возникает с ее распространением и продажей. И ничего удивительно в этом нет: готовый бизнес с неба не падает. Нужно изрядно попыхтеть, прежде чем плод бессонных программерских ночей начнет приносить прибыль.

Готовых и стопроцентных рецептов здесь, в общем, нет. Как и в любом другом торговом бизнесе, в продаже shareware-программ всячески приветствуются инновации и изюминки, которыми ты можешь привлечь покупателя. И это в первую очередь касается даже не методов раскрутки и PR, а качества исполнения самого софта.

ВСТРЕЧАЮТ ПО ОДЕЖКЕ...

Прежде чем браться за распространение программы, десять раз подумай, а стоит ли? Я, конечно, не буду отрицать, что при умелом подходе можно продать отстой в красивой упаковке, но если браться за идею профессионального шароварения, то такие мысли, безусловно, нужно сразу же отмести. Не стоит опускаться до уровня торгаша на рынке, который только и думает, как обмануть и наколоть покупателя. В этой статье я приведу обзор лучших PR-сайтов для шароварщиков, архивов софта, регистраторов. В этих сервисах зарегистрированы сотни тысяч самых разнообразных программ. Среди них не одна сотня твоих конкурентов. И если написанная тобой программа может похвастаться всего лишь парой-тройкой никому не нужных функций, то знай: грош ей цена. Такая серая мышка, скорее всего, даже не попадет в поле зрения покупателя. Что там говорить о возможности ее продажи...

Отсюда первостепенная установка: подготовь программу к продаже. Доведи ее до ума, особенно в плане функциональности и работоспособности, для чего потребуется небольшое маркетинговое исследование: прощупай программы конкурентов и реализуй все то, чего не хватает твоей программе. С конкурентами нужно конкурировать, в этом и есть смысл торговли. Необходимо постоянно пытаться переплюнуть оппонента, предоставить пользователю побольше возможностей или выполнить их на более высоком, чем у конкурентов, уровне, обеспечить максимально возможное быстродействие.

Практика показывает, что можно продать абсолютно любую программу: пускай тот же аудиоконвертер, у которого на рынке тысяча и еще один конкурент. Отличный способ повысить продажи программы - добавить в нее несколько уникальных возможностей. Это не только дополнительный способ привлечь покупателя к покупке программы, но еще и то, на чем можно сделать акцент PR-компании. Аудиоконвертеру, к примеру, никогда не помешает встроенный редактор тегов и функциональный катализатор, предназначенный для автоматической сортировки обработанных файлов по нужным директориям. А еще лучше добавить возможность вставить в аудиофайл свою звуковую вставку так, чтобы пользователь таким образом мог обозначить свои копирайты. И после этого ты можешь раскручивать свою программу как изумительное универсальное средство, которое подойдет как рядовому домашнему пользователю, так и владельцу крупного медиаресурса.

Грамотное оформление - еще один способ добиться успеха. Даже самая простая программа может привлечь внимание покупателя, если она качественно и приятно оформлена. С точки зрения торговли раскрутить и продать такую программу намного проще, чем мощную софтину с уникальными возможностями и быстрыми алгоритмами, но имеющую убогий интерфейс. Внешний вид утилиты должен быть в меру броским, чтобы порадовать установившего программу человека, но при этом не испугать слишком пестрым оформлением. При этом не стоит увлекаться и забывать об интуитивной понятности интерфейса или, как это сейчас модно называть, usability. Подробнее об этом аспекте читай в соответствующей статье номера.

ЧТО В ИМЕНИ ТЕБЕ МОЕМ...

Программа должна иметь хорошо звучащее название. Для "молодых" проектов, в которых какая-то брендовая политика просто отсутствует, оно должно быть еще и информативным. Суди сам. Возьмем, к примеру, почтовую программу всех времен и народов The Bat! ("летучая мышь!", кратко и мажорно). Это имя знает каждый, но не только потому что это чрезвычайно полезная и функциональная программа. В ее раскрутку было вложено немало денег, которой занимались, как я понимаю, профессионалы.

А теперь вернемся к нашему гореаудиоконвертеру. Его, разумеется, можно назвать ястребом (The Hawk), что будет символизировать высокую скорость процесса преобразований. Но смысл? Едва ли каждый пользователь, воспользовавшись своими скрытыми телепатическими способностями, без каких-либо проблем догадается о ее назначении... Название должно давать пользователю как можно больше информации о программе. Fast Audioconverter в этом плане круче, но тоже не фонтан :). Если оригинальности у самого не хватает, обращайся к креативным друзьям и знакомым. Имя, как правило, выбирают раз и навсегда.

Стоит заметить, что программы, ориентированные на отечественный рынок, в идеале должны иметь русское название. "Домашние финансы", "1С: Бухгалтерия" - знакомые слова для нашего потребителя. Впрочем, если быть откровенным, то заработать деньги на шароварных программах во многом проще на западном рынке, чем на нашем, где люди в большинстве своем привыкли к халяве и, более того, никогда не имели дело с покупками в Сети.

Помимо всего прочего, для грамотного PR'а программы жизненно необходим официальный сайт, причем не простая домашняя страничка с парой ссылок на закачку и кнопкой "Купить". Ни в коем случае! Это должен быть полноценный и качественно оформленный сайт. Такой, чтобы у посетителя сложилось впечатление, что он попал на сайт профессионалов. Если ты будешь заниматься разработкой программы в одиночку, афи-

шировать этого не стоит: часто пользователи к этому относятся весьма скептично. Представься группой разработчиков - хуже от этого не будет точно.

На сайте нужно доходчиво объяснить, как работать с программой, какие возможности она предлагает, чем лучше своих конкурентов (можно даже привести таблицу со сравнением характеристик). Не лишним будет предложить интерактивный тур по программе или, по крайней мере, выложить несколько ее скриншотов в хорошем разрешении. Короче говоря, сделай абсолютно все, чтобы пользователь запал на твою программу еще не увидев ее в работе. Впоследствии, когда твой продукт уже завоюет некоторую популярность и получит награды файловых серверов (об этом еще расскажу), расскажи посетителям сайта о своих успехах. Если о твоем детище напишут в прессе – еще лучше. Смело выкладывай ссылку на такой обзор!

Домен второго уровня со стабильным хостингом – неотъемлемые атрибуты современной программы. В дальнейшем во время регистрации на различных сервисах придется указывать ссылку на сайт повсеместно, и тогда она всегда будет на виду у пользователей. Думаешь, кто-нибудь станет смотреть на сайт, расположенный на narod.ru? Вряд ли...

СОФТ НАПОКАЗ

Итак, все приготовления сделаны, самое время начинать PR-кампанию. Стоит начать с регистрации продукта в специализированных софт-архивах. Такого рода сервисы специально предназначены для хранения информации о тысячах программ. Само собой разумеется, это сделано не ради прикола. Толпы пользователей ежедневно посещают такие ресурсы, чтобы найти для себя что-нибудь полезное. Каждый из них теоретически может заметить твою программу, но чисто теоретически. На практике такие программы замечают единицы. Чаще всего вновь прибывшая в софт-архив программа просто теряется среди сотен подобных ей, однако в первые дни появления продукта шансы на его успех максимальны. Практически все сервисы помечают новые поступления специальным хорошо заметным значком - твоя программа тоже этого достойна. Мало того, в таких сервисах часто есть разделы, куда помещается вся информация о новинках дня/недели/месяца. Именно в это время твоя программа будет наиболее заметна. Естественно, нужно суметь выделиться еще и среди новинок, что тоже весьма и весьма непросто.

Для этого ты должен составить программе четкое, яркое и завлекающее описание. Объем такого послания пользователю, как правило, жестко ограничен, поэтому не пытайся уместить в 500 байтах информацию обо всех функциях. Расскажи только о самых вкусных, самых полезных и уникальных. Иногда софт-архив предоставляет возможность дополнительно дать еще и полное описание: вот здесь можно оторваться и описать все, что ты захочешь. Важная хитрость для публикации самой первой версии своей программы - не торопиться с выставлением клейма "Версия 1.0". Пользователи довольно подозрительно относятся к таким продуктам, что вполне логично. Оно и понятно: незнакомая программа может показать во всей красе свою неотлаженность, напортить систему и сделать еще кучу нехороших дел...

Каждая зарегистрированная в каталоге утилита должна пройти контроль web-мастера или модератора. Если она ни на что не годится, ее скорее всего не пропустят. Если же, наоборот, представляет собой сокровище, то есть все шансы, что ее отметят какой-нибудь наградой. Например, дадут пять звездочек, титул "Выбор редакции", "Хороший продукт" или что-то подобное. Такой поворот событий тебе, естественно, на руку. Во-первых, значительно возрастет количество загрузок (и покупок) программы. А во-вторых, будет чем убедительно доказать престижность и профессиональность программы на официальном сайте. Чтобы увеличить шансы на получение подобной награды, лучше безвозмездно предоставить администрации и работникам сервиса бесплатные регистрации :).

Софтовых архивов нынче не просто много, а очень много. Некоторые из них (меньшинство) очень популярные, но платные. Другие совершенно бесплатны, но количество их посещений оставляет желать лучшего. Я рекомендую использовать и те, и другие, хотя все, безусловно, зависит от твоих собственных соображений.

www.download.com

Один из самых старых, известных и популярных сервисов. Поместив сюда программу, можно быть уверенным на все 100%, что она будет скачана не раз и не два, а несколько тысяч раз. Неплохо, да? Но есть проблема: сервис платный. Абонентка на месяц на стандартный пакет услуг составляет девять зеленых енотов. Можно сэкономить заплатив 99$ сразу за год. Стандартный пакет позволит разместить краткое и полное описание программы, ее скриншоты, ссылку на закачку и кнопку "купить". Предусмотрена возможность бесплатного размещения дистрибутива программы на их серверах с помощью дружественного сервиса www.upload.com. Это особенно актуально, если закачки твоего продукта сильно нагружают канал хостера. Помимо этого будет реальная воз- »

www.download.com - ресурс первостепенной важности.

Цена программы тоже во многом влияет на объемы продаж. Лучше в этом деле ориентироваться на цены конкурентов, чтобы и палку не перегнуть, но и меньше заслуженного не получить. Дорогая программа никому не нужна, а слишком дешевая вызовет подозрение у покупателя.

Важно убедить покупателя, что ему предлагают качественный продукт, а не кота в мешке. Покупатель должен быть уверенным, что в нужный момент программа не откажет, а если даже и произойдет сбой, то разработчик поможет в самые короткие сроки. ICQ-поддержка и форум на сайте - главные орудия такого убеждения.

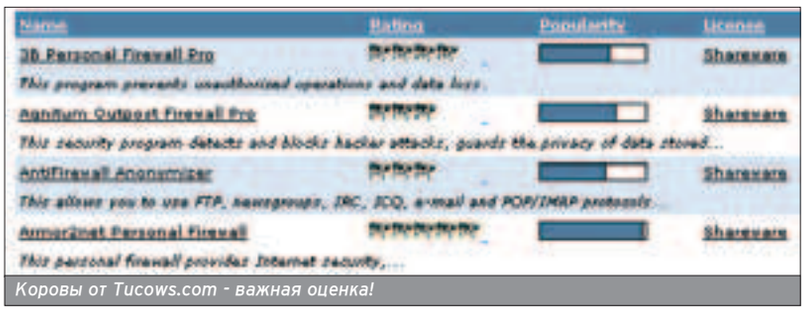
Во время регистрации в софт-архивах можно значительно облегчить себе жизнь с помощью так называемых PAD-файлов. Что они представляют собой? PAD (Portable Application Description) - это специальным образом оформленный XML-файл, который содержит всю необходимую информацию о твоей программе: описание, размер, номер версии, путь к установочному файлу, URL скриншота и т.д. Большинство софт-архивов на ура импортируют такую информацию, а значит, тебе не придется вводить свои данные по двадцать раз на дню: достаточно указать путь к заранее подготовленному PAD'у, тем более что создать его - раз плюнуть. Для этого были разработаны специальные программы. Среди них PADKit (**www.padkit.org**) и PADGen 2 (**www.padgen.org**).
Более подробную информацию можно найти на сайте Ассоциации профессиональных шароварщиков - **www.asp-shareware.org**.

можность создать распределенную систему зеркал, с которых посетители смогут выкачать программу. Возвращаясь к www.download.com, отмечу, что за определенную денежку можно поместить свою программу на верхних строчках каталога и платить сервису за каждое скачивание. Каждой программе, независимо от выбранного пакета услуг, присваивается несколько ключевых слов. Рекомендую отнестись к их выбору сознательно, серьезно и на трезвую голову – от этого во много зависит эффективность сервиса.

www.tucows.com

Так называемый "коровий" сайт :). Этот легендарный софт-архив примечателен тем, что рецензирует каждую присланную ему программу, а результат проверки выводит в весьма оригинальных условных единицах - в коровах. Четыре коровы – программа хорошая, три - так себе. Оценка в пять коров фактически означает гарантию качества продукта. Ресурс ежедневно принимает сотни заявок от производителей самого разного программного обеспечения (не только для компьютеров, но и для смартфонов, и для КПК). Ясное дело, никто не будет браться за бесплатное рецензирование такого количества софта, поэтому услуги www.tucows.com стоят недешево. Чтобы разместить свою софтину, необходимо сначала зарегистрироваться в так называемом центральном ресурсе разработчиков ARC (Author Resource Center). В дальнейшем с помощью этого ресурса ты сможешь добавлять новые программы, редактировать описания имеющихся. По окончании ввода характеристик утилиты ресурс подсчитает срок, через который твоя заявка будет обработана. На скорое исполнение своих желаний рассчитывать не стоит – месяц как минимум. За отдельную плату можно сократить срок ожидания до недели или даже до одного дня. Но сам понимаешь, все это стоит недешево, тем более что за само размещение программы нужно тоже выложить круглую сумму. Все-таки такие затраты оправдываются. Кстати, подобного рода услуги можно оплатить только с помощью кредитных карт Visa, MasterCard/Eurocard and American Express. Другие платежи, как и на download.com, не принимаются.

Позволь представить – коровий сайт (www.tucows.com)



Коровы от Tucows.com - важная оценка!

Почти все регистраторы высылают деньги двумя способами: чеком и банковским перечислением. Перечисления идут быстро, но банк берет большую комиссию. Более того, такого рода переводы могут попасть в поле зрения налоговой полиции, а с ней, как известно, шутки плохи. Чеки идут долго, но, с другой стороны, ты практически ничего не теряешь на пересылке и, что немаловажно, можешь спать сладко и видеть радостные сны :).

www.freeware.ru, www.download.ru, www.softodrom.ru...

Это наши родные софт-архивы. В России платить деньги не любят, поэтому эти сервисы за свои услуги денег не требуют, но и пользы от них, прямо скажем, немного. Прибегать к их услугам стоит если продукт ориентирован на отечественного потребителя, а иначе их можно применять только как довесок к основной PR-кампании. Да и перечень услуг не такой шоколадный. Например, многие из таких сервисов не предоставляют хостинг под исполняемые файлы твоей программы, другие частенько пребывают "в ауне" и т.д. Однако при всех минусах не стоит забывать об их главном достоинстве - бесплатности. Тем более что процесс добавления программ в русские каталоги можно выполнить всего за несколько минут с помощью специальных программ наподобие Put Soft (www.chipmicro.com/rus).

ДРУГИЕ СПОСОБЫ PR'А

■ Одной регистрацией в софт-архивах значительных результатов не добьешься. Следующий шаг к успеху - работа с поисковыми системами. Причем ориентироваться нужно не на отечественных грандов типа www.yandex.ru, www.rambler.ru и т.п. (хотя и это тоже не помешает), а на популярные западные поисковики, прежде всего www.google.com, www.yahoo.com, www.altavista.com.

Задача непростая: нужно всячески постараться сделать так, чтобы при любом запросе, связанном с тематикой программы, твоя ссылка была на первой странице. Чем выше, тем, естественно, лучше. Не нужно объяснять, что добиться этого чрезвычайно сложно и, по правде говоря, в некоторых случаях даже невозможно, но попытаться определенно стоит, тем более что в Сети широко распространены материалы, которые доступно и подробно излагают теорию оптимизации. Самые толковые из них – сайт searchengines.ru и архив рассылки www.optimization.ru/subscribe/list.html.

Каждая поисковая система имеет свои собственные алгоритмы поиска и сортировки полученного результата, но для каждой из них актуально правило: чем выше индекс цитируемости ресурса, тем выше его месторасположение на странице с результатами поиска. В свою очередь индекс цитируемости напрямую зависит от того, насколько часто ссылка на ресурс фигурирует в других индексируемых поисковиком сайтах. Понимаешь, куда я клоню? От тебя требуется, чтобы имя программы встречалось на просторах Сети везде и всегда, чтобы твой продукт попал в новости самых разных сайтов software-тематики, конференций и форумов, пускай даже вареных. Даже если пара тысяч человек скачает твою программу со свежим кряком, то особенно хуже тебе от этого не станет: они бы ее все равно, скорее всего, не купили... Зато имя твоей программы засветиться для роботов

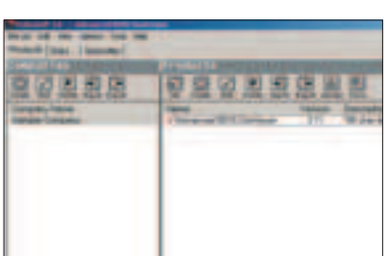
Один из популярных российских софт-архивов

www.shareit.com – один из самых старых регистраторов

поисковиков и, что еще важнее, для пользователей.

После того как ты самостоятельно зарегистрируешься в ведущих поисковых системах, можно подумать и о мелких сошках. Вручную заниматься этим не рекомендую (угрохаешь кучу времени), но и обращаться к занимающимся раскруткой конторам не стоит. Идеально будет воспользоваться специальной софтиной. Одна из них - Page Promoter (netpromoter.ru/pagepromoter) от русских разработчиков.

Еще один способ набрать себе очки – оплатить контекстную рекламу, которую сейчас предлагает любая поисковая система. Принцип ее действия невероятно прост: ты кладешь на свой счет некоторую сумму денег, выбираешь себе несколько ключевых слов и указываешь их поисковику. Отныне во время подходящего запроса ссылка на твой сайт будет отображаться в специально отведенном и выгодном в плане привлечения внимания месте. Оплата подобного рода рекламы в большинстве случаев осуществляется по факту перехода посетителя по ссылке. Денег с тебя много не возьмут, зато эффект от такой рекламы колоссальный. Она куда эффективнее, чем та же баннерная реклама, которая хотя и уместна, но едва ли способна так повысить эффективность.



RoboSoft (www.rudenko.com) – программа для автоматической регистрации в бесплатных софт-архивах

А КАК ЖЕ ДЕНЬГИ?

Итак, мы дошли до самого интересного - до непосредственной продажи ПО. Прошли те времена, когда пользователи открывали счета в банке и покупали мерчант-аккаунты, чтобы самостоятельно принимать платежи. Сейчас этот процесс целиком взвалили на так называемых регистраторов, своеобразных посредников между продавцом и покупателем. Такие сервисы самыми разнообразными способами (прежде всего посредством кредитных карт) принимают платежи покупателей, а потом, взяв себе определенный процент, пересылают деньги продавцу.

www.regnow.com

Один из самых популярных регистраторов, который давно завоевал доверие как продавцов, так и покупателей ПО. Принимает все виды платежей, при этом способен генерировать и отсылать регистрационные ключи. Себе в карман берет 16% (минимум $2) от каждой сделки, но плюс к этому тебе придется единожды заплатить $9 за регистрацию в системе. В свою очередь сам сервис стабильно высылает продавцам деньги два раза в месяц: переводом на банковский счет или чеком. Примечательно, что с помощью www.regnow.com ничего не стоит наплодить себе кучу аффилиатов (перепродавцов твоего ПО), что поможет повысить продажи еще больше.

www.shareit.com

Еще один проверенный временем регистратор. Поддерживает два вида комиссии: $2,95+5% с каждой сделки или просто 14,9% (но не меньше $2,5). При этом за регистрацию в системе плата не взимается. Моментально обрабатывает платежи с кредитных карточек и без задержки высылает лицензионные ключи, за что полюбился многим покупателем. Генератор ключей для сервиса нужно оформить особым образом, используя их собственный SDK (Software Developer's Kit). Поставляется он с документацией, поэтому проблем возникнуть не должно. По достижению некоторого минимума на счету www.shareit.com вышлет деньги с помощью чека или банковским переводом. Система имеет интерфейс на нескольких языках и работает с несколькими валютами одновременно.

www.softkey.ru

Незаменимый регистратор для тех, кто ориентирует свой продукт на российский рынок. Пожалуй, единственный полноценный регистратор во всей России. Примечателен тем, что в качестве способа оплаты помимо кредитных карточек принимает переводы через "Сбербанк", а также посредством различных электронных платежных систем (Webmoney, Yandex.деньги и т.п.). Все это, конечно, хорошо, но заработать приличные деньги с этим регистратором сложно, а все потому что www.softkey.ru берет 32,2% от стоимости программы, если автор не является плательщиком НДС. Считай сам.

Если собираешься ориентировать свой продукт на российский рынок, то тебе прямая дорога сотрудничать с www.softkey.ru

Работать со всеми регистраторами одновременно не имеет никакого смысла. Идеально сотрудничество с двумя или, максимум, с тремя, да и то на случай если один из них внезапно "упадет". Помимо перечисленных регистраторов, в Сети работает еще не один десяток аналогичных. Однако работать с кем попало не рекомендую, хотя, конечно, можешь попробовать: полная, но несколько устаревшая сравнительная характеристика регистраторов лежит по адресу mini.net/pub/sharegs.html.

Регистратор www.regnow.com – наш выбор!

Виталий Конр (conr@gmail.com)

ОБЩЕСТВО - ДРУГ ПРОГРАММИСТА

ОБЗОР РОССИЙСКОГО SHAREWARE-СООБЩЕСТВА

Казалось бы, ну что может быть проще - написал программу, выложил на каком-нибудь narod.ru и лежи себе на диване, греби деньги лопатой. Программа-то нужная, всем полезная, но... Без помощи профессионалов ты вряд ли добьешься успеха на этом поприще. Поговорим о российском shareware-сообществе.

Любые профессионалы одержимы тягой к образованию клубов, сообществ, тусовок - называть можно как угодно, суть от этого не меняется. Русских shareware-программистов тоже постигла эта участь.

SWRUS

Расшифровывается эта аббревиатура как Shareware Russia mail list. Строго говоря, называть SWRUS сообществом не совсем правильно, так как изначально это просто лист рассылки (aka группа) на сервере yahoogroups.com. Созданный в 1998 году Виктором Ижикеевым, SWRUS в то время имел всего несколько участников, которые практически все знали друг-друга лично. Первый публичный анонс появился в FIDO'шных сетях 17 февраля 1998 года, с тех пор этот день считается "официальным" днем рождения SWRUS. За семь лет ситуация, естественно, очень сильно изменилась. Лист стал массовым (на сегодняшний день он объединяет около 2000 подписчиков) и обрел уже все качества узнаваемого бренда, причем, что показательно, не только в России. Пока подписаться на него может любой желающий, денег за это не берут. По всем вопросам подписки нужно обращаться к действующему модератору (на момент написания обзора это был Юрий Герасимов, автор Chameleon Clock). Единственное, но очень важное условие - честно заполнить профиль подписчика на сервере Yahoo. Ссылка для подписки http://groups.yahoo.com/group/swrus/join. Кроме основного листа, есть еще несколько дочерних листов, каждый со своей специализацией. О них можно узнать на официальном сайте SWRUS - www.swrus.com.

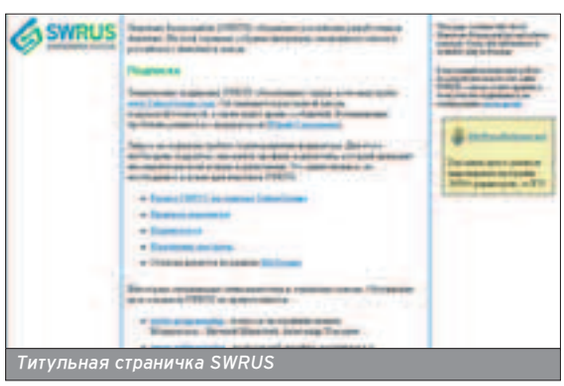
Однако не стоит только подписавшись на эту рассылку тут же начинать задавать там кучу вопросов. Большинство из них уже давно заданы и при очередном их появлении старожилы начинают потихоньку звереть :). Так что лучше, дабы не портить заранее отношения с остальными подписчиками, прочитать SWRUS-FAQ, составленный Александром Лысковским. К сожалению, некоторые ответы уже потеряли свою актуальность (сейчас, например, сервисов для shareware-программистов значительно больше, изменились условия у регистраторов и т.д.), но тем не менее, это один из самых полезных источников в рунете на эту тему. Ознакомиться с ним можно по адресу www.shpaga.ru/faq2.

Кроме того, на сайте SWRUS можно найти каталог сервисов, которые помогут перевести программу на хороший английский, сделают сайт, пропишут его в каталоги и напишут пресс-релиз. В общем, очень полезный для любого shareware-программиста портал, который к тому же с некоторых пор довольно регулярно обновляется.

2. ISDEF

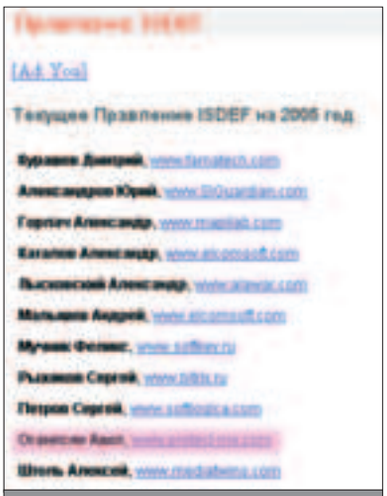
Independent Software Developers Forum - форум независимых разработчиков программного обеспечения. Фактически это первая в России и СНГ ассоциация shareware-программистов. И хотя очень многие ее члены были или являются подписчиками SWRUS, ничего общего друг с другом они не имеют. В отличие от SWRUS, ISDEF - это действительно официально зарегистрированная ассоциация, которая является юридическим лицом и имеет свой устав, правление, банковские счета и членские взносы. Да-да, именно членские взносы, так как бесплатной она не является (еще одно немаловажное отличие). Впрочем, об этом чуть ниже.

ISDEF была создана в 2002 году, официальный сайт - www.isdef.org. В том же году с 28 по 30 сентября была проведена первая массовая акция ISDEF - конференция ISDEF'2002, прошедшая в Черноголовке. Основная цель конференции - "обсуждение вопросов развития и ведения shareware-бизнеса, проблемы развития интернет-маркетинга, защиты авторских прав и многое другое". Хотя, на мой взгляд, самое главное было обкатать

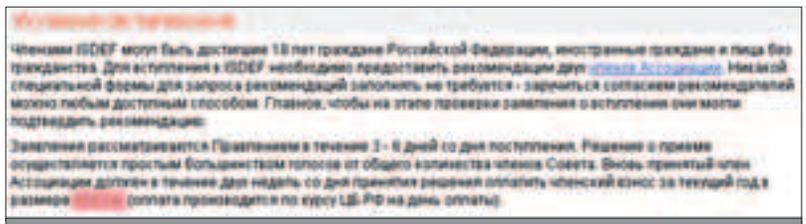


Титульная страничка SWRUS

ISDEF, в отличие от SWRUS, является полноценным юридическим лицом



Один из членов правления ISDEF - полный тезка редактора Спеца :-)



За членство в ISDEF надо каждый год платить 100 у.е.

механизм проведения подобных конференций, так как до этого ничего подобного не наблюдалось, соответственно, и опыта проведения столь сложных мероприятий никто из организаторов не имел. Следует признать, что конференция получилась на редкость удачной, в пользу этого утверждения говорит хотя бы тот факт, что если спонсорами первой конференции выступали фактически сами организаторы (Elcomsoft, TamoSoft, SoftKey и т.д.), то в следующей конференции ту же роль играли такие гиганты IT-бизнеса, как Microsoft и Intel. С тех пор конференции шароварщиков проводятся каждый год, и, что не может не радовать, с каждым годом они становятся все крупнее и интереснее.

Основными своими целями и задачами ISDEF постановила "всяческое развитие организации разработки и внедрения программного обеспечения". На практике, кроме проведения конференций, это выражается, например, в том, что ISDEF поддерживает отношения с такими компаниями, как ShereIt, RegSoft, RegNow, и, в случае возникновения каких-нибудь проблем или спорных ситуаций, может помочь своему члену в разрешении вопроса. Так, в частности, при регистрации ShareIt проверяет данные автора и во время этой проверки продавать программы через этого регистратора невозможно, да и гарантии того, что проверка будет пройдена, нет - заявка может быть отклонена. Если же есть поручительство ISDEF, то проблема отпадает сама собой, так как проверки попросту не будет и аккаунт активируют сразу же.

Членом ISDEF может стать любой человек, дорослий до 18-ти лет, вне зависимости от гражданства, или юридическое лицо (впрочем, этот вариант для нас не важен, так что я его пропущу). Но все не так просто, подвох есть всегда! Если на SWRUS может подписаться фактически любой желающий, о котором никто из подписчиков скорее всего ничего не знает, то для того чтобы тебя приняли в ISDEF, нужно предоставить рекомендации двух действующих членов этой ассоциации. Если рекомендации есть и Советом будет принято положительное решение о приеме (обычно принимается в течении трех-пяти дней), новичок должен в течении двух недель оплатить членский взнос за текущий год в размере $100 по курсу ЦБ на день оплаты. Естественно, этот взнос не разовый, а ежегодный, причем если его вовремя не оплатить, то происходит автоматическое исключение из рядов ассоциации. Исключенный член или по каким-то причинам не принятый кандидат может подавать заявление на вступление не ранее чем через год, а если такое заявление было отклонено два раза, то больше оно рассматриваться никогда не будет, и о перспективе быть членом ISDEF придется забыть. Так что не стоит сразу же после прочтения статьи бежать писать заявление - особой пользы не будет :).

ВСТУПАЕМ В ISDEF

Если есть знакомые shareware-программисты, члены ISDEF, которые согласны дать свои рекомендации, то все очень просто. Нужно всего лишь заполнить форму подачи заявления на сайте ISDEF и дожидаться решения совета. Если было принято положительное решение, то оплатить членский взнос, и все - ты уже в ISDEF. Об отрицательном решении думать не будем, так как подавать заявление имеет смысл только в том случае, если есть свой продукт, который продается, а в рамках SWRUS становится тесно и хочется большего. В этом случае отказ, думаю, маловероятен.

Что же делать? Не то что рекомендаций нет, а даже нет ни одного знакомого, который был бы членом ISDEF? Ответ очень прост и одновременно сложен. Нужно вспомнить об SWRUS - как было сказано выше, многие члены ISDEF одновременно являются подписчиками SWRUS, и более того, любой член может читать конференцию SWRUS через news-сервер ISDEF. Так что вполне можно "засветиться" для начала в SWRUS, и если твоя программа действительно стоящая, то есть очень неплохие шансы заинтересовать кого-нибудь из членов ISDEF. Многие из них все еще с ностальгией вспоминают те времена, когда сами были новичками, и вполне могут пойти навстречу талантливому неофиту. Пусть этого добиться нелегко, но вполне осуществимо, так что желаю удачи.

ЗАЧЕМ ЭТО НУЖНО?

Зачем вступать в какие-то сообщества, что-то кому-то доказывать, искать рекомендации? Да, в общем-то и незачем. Никто тебя не заставляет. Просто, как известно, на чужих ошибках учиться куда приятнее, чем на своих, а то, что учиться придется, - это факт. Причем учиться постоянно, не прекращая ни на минуту, потому как область информационных технологий меняется очень быстро и то, что было откровением сегодня, завтра будет никому не нужно. И потом, это же просто-напросто интересно. Например, вполне вероятно, что на следующей конференции ISDEF будет Кевин Митник, разве не интересно пообщаться со столь известной личностью вживую? Более того, 12 февраля уже состоялась встреча известного хакера с членами ISDEF. Решать тебе.

Чтобы тебя приняли в ISDEF, нужно предоставить рекомендации двух действующих членов этой ассоциации.

12 февраля состоялась встреча членов ISDEF с Кевином Митником

Денис Колисниченко (dhsilabs@mail.ru)

ЗАЩИТИ СЕБЯ САМ

ТЕХНОЛОГИИ ЗАЩИТЫ ПРОГРАММНЫХ ПРОДУКТОВ

Ты написал программу и хочешь получить за нее материальное вознаграждение, поскольку морального вроде "Отличная программа, спасибо!" не всегда бывает достаточно. Как защитить свою программу от несанкционированного использования? В этой статье поговорим о нестандартных решениях, позволяющих защитить твои программы.

ДЕМО-ВЕРСИЯ

Практически любую систему безопасности, разработанную одним человеком, другой может взломать. Почему? Потому что если один человек сделал что-то, наверняка найдется еще один, который сможет повторить это с точностью до наоборот. Кстати, о дизассемблировании программ мы тоже поговорим, только попозже. Предположим, ты выложил в интернет "защищенную" версию своей программы. Например, программа в одном из файлов, нарочно "замаскированном" под DLL, хранит счетчик запусков. Как только счетчик превысил число 60 (или любое другое), выводится сообщение о том, что пора бы уже заплатить за программу, и появляется окошко с полями ввода регистрационного имени и серийного номера.

Теперь попробуем проанализировать действия пользователей. А пользователи, как мы знаем, бывают разными. Один сразу же зайдет на твой сайт, свяжется по e-mail или позвонит по телефону. В итоге он обменяет несколько зеленых бумажек на заветный серийный номер. Другой пользователь попробует переустановить программу. В зависимости от твоего алгоритма защиты программа или сбросит счетчик, или сообщит пользователю все о том же: пора купить серийный номер. В первом случае будет сам-знаешь-что: твою программу будут использовать до очередного переформатирования жесткого диска. А во втором... Это зависит от пользователя. Или он попытается найти crack, или просто удалит программу и будет искать ее аналоги. Платить этот пользователь не будет: он бы это сделал с самого начала, разве что программа настолько уникальна, что в мире нет ее аналогов (правда, в этом случае не думаю чтобы она была shareware). Почему не будет платить? А ты как часто платишь за используемые shareware-программы? Думаю, этот вопрос обсуждать больше не будем...

Остался еще один класс пользователей - несдающиеся пользователи. В Сети много программ, позволяющих наблюдать за процессами, а именно: выяснять, какой ключ реестра используют процессы, какие файлы открывают, какие данные они сбрасывают в файл, а какие загружают из файла. Такой пользователь за пару минут вычислит твой файл и обнулит счетчик. Все - твоя программа попала в вечное рабство. А если пользователь еще и не жадный, он напишет в Сети о том, как он ловко взломал твою программу. Что в итоге? Все пользуются твоей программой, а ты так ничего за нее и не получил.

Вот для этого нужна демо-версия, с помощью которой ты хоть что-нибудь да получишь - теорема такова, что "найдется хотя бы один пользователь, который заплатит за твою программу" (хорошо хоть не "один и только один"). Сейчас попробуем ее доказать.

Предположим, ты написал программу, формирующую бланк налоговой накладной (или любой другой бланк). Пользователь создал накладную, нажимает кнопочку "Печать", а ему в ответ "Введите серийный номер, иначе печатать не будем". Пользователю платить деньги не хочется. Он каким-то образом подбирает этот самый номер, и бланк выводится на печать. А теперь представим, что это демо-версия и в ней функции печати нет изначально. Будь у пользователя хоть десяток серийных номеров, ничего распечатать он не сможет. Выход один - связываться с тобой и заказывать полноценную версию. Как правило, демо-версии даже не взламываются - взламывать там нечего.

Конечно, вся описанная ситуация немного надумана. Если твоя программа популярна, то в Сети уже есть полноценная взломанная версия, которую можно использовать, если демо-версия понравилась. Но несколько первых пользователей все равно заплатят деньги. И будь уверен: сами эти пользователи ни за что не опубликуют где-то на сайте их собственные версии, потому что сработает "Я ж за нее деньги заплатил!". Другое дело, если у них ее кто-то может украсть или они смогут ее продать за полцены. В этом случае нужно придумать такой алгоритм, который бы препятствовал запуску программы на другом компьютере.

СЕРИЙНЫЕ НОМЕРА

Вот мы только и говорим о серийном номере. А как написать эффективный алгоритм генерирования серийных номеров, чтобы номер было трудно подобрать? Если ты не математик, лучше воспользоваться какой-нибудь программой, позволяющей управлять серийными номерами, чем писать "очень сложный" алгоритм, который в качестве серийного номера принимает сумму кодов символов введенного регистрационного имени... Говорю "Не нужно так делать", но найдется хотя бы один читатель, который подумает: "Отличная идея!"... Еще раз повторяю: вместо такого алгоритма воспользуйся какой-нибудь программой. Понимаю, что стандартные решения стандартно и взламываются, но для первой версии твоей программы этого будет достаточно. Одну из таких программ мы рассмотрим в этой главе, а пока поговорим о дизассемблировании.

ЗАЩИЩАЕМ АЛГОРИТМ

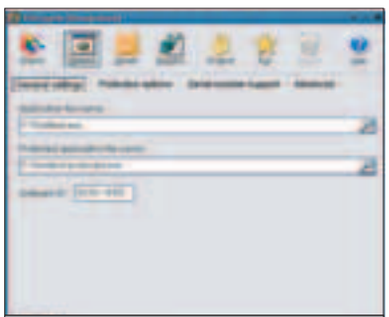
Ты же не хочешь, чтобы твой алгоритм генерации был взломан за пару часов с помощью какого-нибудь дизассемблера. Как это так получается? Есть такая целая наука - reverse engineering, по-нашему - обратная разработка (проектирование). Ты написал программу, компилятор ее откомпилировал - на выходе exe-файл, как обычно. Другой человек запускает дизассемблер (или какой-нибудь отладчик вроде SoftICE) и буквально "по косточкам" разбирает твою программу. Ясно, что сам исходный код он не увидит (впрочем, это зависит от опций компилятора), то есть он увидит не If Serial.Text= ..., а то, что ему нужно. Поверь, если человек этим занимается, он знает, куда смотреть. По су-

ти, он проходит обратный путь: его исходный код – это твой exe-файл. Отсюда и название – обратное проектирование. Чтобы твоя программа была недоступна для большинства отладчиков и дизассемблеров, используй специальные программы, изменяющие exe-файл так, что его невозможно дизассемблировать. Защитив таким способом свою программу, сможешь спать спокойно – твой алгоритм никто не узнает.

EXECRYPTOR

EXECryptor – это специальная программа, позволяющая с минимальными затратами времени решить все вышеизложенные проблемы: она и серийный номер придумает, и программу защитит от отладчиков. Функции программы внушают доверие:

- метаморфическое преобразование кода программы, позволяющее защитить программу от дизассемблирования и модификации;
- защита ключом отдельных участков кода программы (поддерживается только в зарегистрированной версии);
- полное разрушение логики защищенных фрагментов кода, не позволяющее анализировать программу с помощью дизассемблера или отладчика;
- обнаружение и противодействие отладчикам SoftIce, NtIce, TD и др.;
- защита точки входа;
- защита от модификации кода;
- защищенная работа с реестром, не позволяющая программам вроде RegMon определить, к какому ключу реестра обращается твоя программа;
- технология "динамического импорта", которая разрушает имена всех импортируемых функций, а также не использует функцию GetProcAddress;
- сжатие ресурсов и исполнимого кода приложения;
- поддержка коротких серийных номеров (12 символов);
- поддержка внешнего генератора серийных номеров с OLE/DLL-интерфейсом;
- технология OneTouch Trial (о ней читай ниже).



EXECryptor решит много проблем

EXECryptor позволяет защитить любое 32-разрядное PE-приложение (.exe, .dll, .bpl, .vxd, .wdm). Программа тестировалась с операционными системами W9x/ME/NT/XP/2003. Кстати, доступно SDK для Delphi, C++ Builder, MSVC и MSVB.

А теперь поговорим подробнее о некоторых функциях программы. Самое главное, что нас интересует – это метаморфическое преобразование кода программы и поддержка серийных номеров. Метаморфическое кодирование позволяет изменить код твоей программы до неузнаваемости и запутать отладчик и человека, который запустил этот отладчик. После такого у этого человека сразу пропадет желание взламывать твою программу. Смотри, вот исходный код оператора, выводящего строку:

```
writeln('Test OK');
```

Вот что получается после компиляции:

```
mov eax, [$004092ec]
mov edx, $00408db4
call @WriteOLSTring
call @WriteLn
call @_IOTest
```

Даже человеку, не особо знакомому с процессом дизассемблирования, понятно, что сейчас происходит – выводится какая-то строка. А теперь применим к тем четырем строчкам метаморфическое преобразование кода:

```
xchg [edi],dl
db 3
add al,$30
xlat
call +$0000025b2
jmp +$00000eec
call +$00000941
or al,$4a
scads
call -$304ffbe9
rol eax,$14
mov edi,[ebx]
jmp +$000001738
mov ebx,eax
```

»

МНЕНИЕ ЭКСПЕРТА

Евгений "Firstborn" Рогов, аналитик и специалист по качеству ПО

Защита программного продукта от взлома – задача совершенно нетривиальная. Для того чтобы хоть немного сбить с толку опытного взломщика, потребуется немало усилий разработчика ПО. Как правило, таких созидательных усилий требуется приложить в несколько раз больше, чем для того чтобы свести всю защиту на нет. В результате, если не забывать о коммерческой рациональности, неизбежно приходим к выводу о том, что в большинстве случаев разработчику shareware-проекта разумнее бросить все силы на доведение до ума собственно ПО и на его раскрутку, чем надолго концентрироваться на создании безупречной защиты, какой вообще-то не существует. В такой ситуации целесообразно присмотреться к имеющимся на рынке системам защиты ПО от взлома – покупка хорошо зарекомендовавшей себя "навесной" защиты может стать полезным капиталовложением, сберечь силы и время разработчика. Однако вне зависимости от того, какой способ защиты программы выбран, рекомендуется регулярно отслеживать ситуацию со взломом защищаемого ПО: время от времени пытаться найти выложенные в Сети украденные серийные номера, ключи активации, генераторы ключей и т.п. Если подобные материалы будут обнаружены, реагировать четко и без промедления, обеспечив неработоспособность имеющихся в наличии средств взлома. Сделать это несложно: как правило, достаточно слегка поменять некоторый параметр алгоритма генерации ключей или внести имеющиеся в свободном доступе ключи в "черный список". И не стоит тратить энергию на попытки убрать из Сети выложенные там ключи и их генераторы, поскольку, скорее всего, эта задача окажется непосильной. К тому же несложно представить себя на месте нечестного пользователя, который, казалось бы, нашел множество ключей для интересующей его программы, но ни один из них не работает. В такой ситуации энтузиазм быстро иссякнет, и искать дальше такой пользователь не захочет, а возможно, даже задумается о покупке программы (впрочем, вряд ли).

```
shr ebx,$03
push ebx
jmp +$00001b5e
call -$000001e8
jmp +$00003203
jmp +$00005df8
call +$000000910
adc dh,ah
fmul st(7)
adc [eax],al
les eax,[ecx+$0118bfc0]
stosb
...
```

Полный список составляют более 500 инструкций. Попробуй разобраться, что тут к чему. К тому же не упоминается ни одно название функции, как в предыдущем примере.

ПРАКТИКА

Хватит слов, на конкретном примере попробуем защитить произвольную программу. Для этого я написал небольшую программку на Delphi - test.exe. Нажимаем кнопку Project в окне EXECryptor, выбираем New, вводим имя проекта и сохраняемся. После этого сразу переходим в раздел Options. На закладке General Settings вводим основные параметры проекта – имя исходного exe-файла и защищенного exe-файла. Параметры защиты кода программы находятся на закладке Protection Options.

Опции защиты

Уровень метаморфического преобразования кода задается бегунком Virtualization level – чем больше, тем лучше, точнее, тем сложнее будет что-то сделать с твоей программой. Не бойся и выбирай стопроцентный - размер exe-файла не увеличится до размеров папки Windows. У меня он даже стал меньше. Конкретные параметры таковы: 1) исходный размер (незащищенная программа) – 292 Кб; 2) защищенная программа 20% - 456 Кб; 3) защищенная программа 100% - 450Кб.

Можно включить опции антиотладчика (у меня незарегистрированная версия, поэтому они все включены, и выключить их я не могу :-)). Их довольно много:

- Patch protection – защита от всякого рода патчей, проверяется целостность приложения;
- Detect inactive tools – позволяет определять установленные в системе, но незапущенные отладчики;
- Kill Registry/File monitors – "убивает" некоторые мониторы реестра и файловые мониторы;
- Debug messages – если обнаружен отладчик, будет выведено сообщение "Debugger detected" и программа завершит свою работу;
- Anti-trace – запрет трассировки программы;
- Protect entry point – защищает точку входа; если ты пишешь программу на Delphi, тебе нужно в список модулей проекта (а не формы!) добавить модуль EXECryptor.pas, который поставляется вместе с EXECryptor, иначе эта функция не будет работать;
- Dynamic import – функция динамического импорта.

Параметры сжатия:

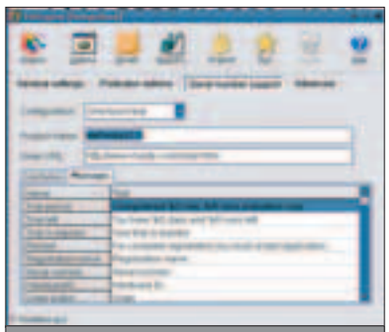
❶. compress resource – сжимать ресурсы, очень полезная функция;

❷. compress code & data – можно сжать код и данные, но это может отразиться на быстродействии твоей программы (особенно снижается скорость запуска);

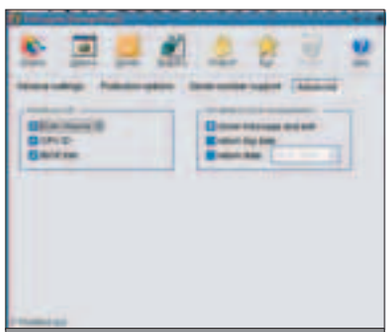
❸. use maximum compression – использовать максимальное сжатие, снижает скорость запуска твоей программы на 10-20%.

Еще одна очень полезная опция, strip relocation info, удаляет из объектного файла таблицу имен и информации о номерах строк. После этого программа не допускает символьной отладки. Помнишь команду strip в Linux – так это все та же функция. Кажется все, программа защищена. Теперь переходим на закладку Serial number support. Первым делом измени режим Configuration. По умолчанию используется значение Manual protection, что означает, что твоя программа сама будет управлять серийными номерами. Выбери режим One-touch trial. После этого установи два самых главных параметра – название твоей программы и URL, где можно получить серийный номер.

На закладке Limitation установи параметры ограничений: количество дней и/или количество запусков программы, это уже на твое усмотрение. Можно включить параметр Allow lock serial to hardware, этот режим позволяет привязывать серийный номер к железу. Когда закончится срок действия, пользователь увидит ID железа, который он будет должен сообщить тебе, а ты уже по этому ID сгенерируешь серийный номер, который будет работать только на той машине. Как это сделать? Подожди, через пару минут узнаешь. На закладке Messages ты можешь изменить параметры окна регистрации: какое сообщение и когда будет выведено.



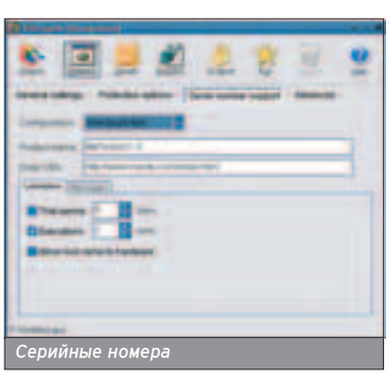
Сообщения окна регистрации



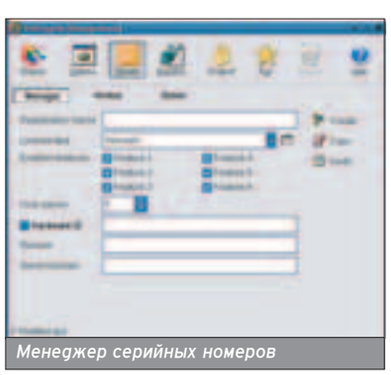
Дополнительные параметры

Закладка Advanced содержит дополнительные опции: будет ли программа реагировать на изменение даты и времени, а также какая информация будет включаться в ID железа (по умолчанию о процессоре, о BIOS и о жестком диске).

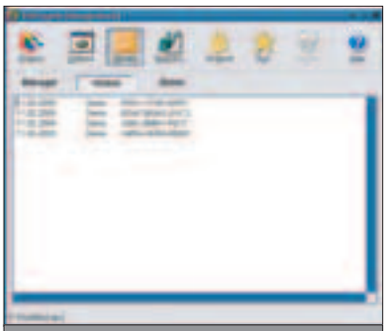
Переходим в раздел Serials. Закладка Manager – это основная закладка, которую ты будешь использовать при создании серийного номера. Вводишь имя пользователя, выбираешь тип лицензии – обычная (Standard) или ограниченная временем (Time limited). В этом случае тебе нужно будет указать дату окончания лицензии. Если ты установил параметр Allow lock serial to hardware, тебе нужно указать Hardware ID – ID железа пользователя. Все, нажимай кнопку Create и в поле Serial number увидишь серийный



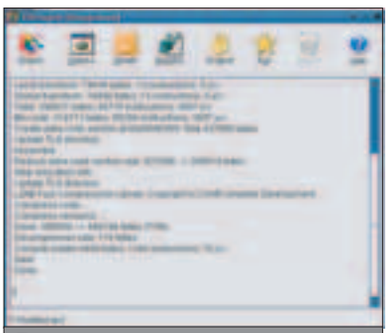
Серийные номера



Менеджер серийных номеров



Созданные ранее серийные номера



Процесс защиты программы

Защищенная программа

номер, который нужно передать пользователю.

На закладке History отображаются созданные ранее серийные номера.

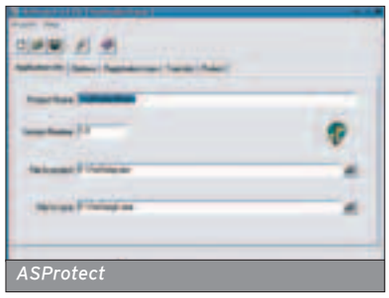
Раздел Registry рассматривать не будем – ты и сам сможешь с ним разобраться. Нажимаем кнопку Protect, ждем пару секунд, и наша программа защищена от взлома и неблагодарных пользователей.

Все, что осталось – запустить программу кнопкой Run. При запуске программы появляется окошко, в котором отображается ID железа и поля для ввода регистрационного имени и серийного номера.

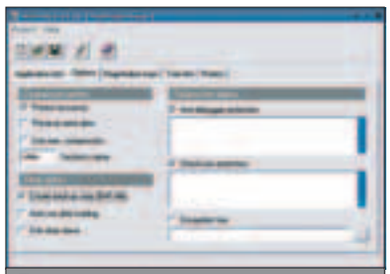
В заключение могу сказать лишь, что если у тебя незарегистрированная версия EXECryptor, по окончании срока (который ты сам установил) твои программы будут выводить сообщение о том, что они защищены незарегистрированной версией EXECryptor.

ASPROTECT

■ EXECryptor – это не панацея от всех бед, то есть не единственная программа такого класса. Есть еще одна удобная программка – ASProtect. Во многом ASProtect и EXECrytor похожи, поэтому очень подробно рассматривать эту программу мы не будем, да и на этих нескольких страницах не поместится.



ASProtect



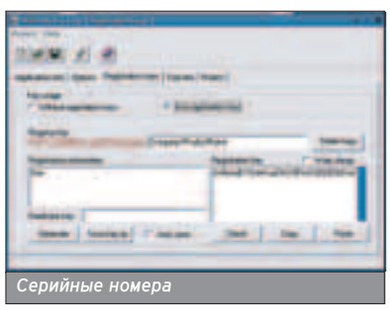
Параметры защиты и сжатия – ASProtect

На закладке Application Info нужно ввести общие параметры: имя исходного и имя защищенного файла, а также название продукта. Закладка Options содержит разные параметры – параметры сжатия, параметры защиты.

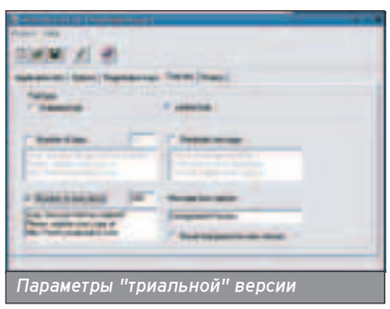
Создать серийный номер можно на закладке Registration Key, а параметры "триальной" версии – на закладке Trial Info. Как и в EXECryptor, ты можешь задать количество дней и/или количество запусков.

Чтобы защитить программу, переходим на закладку Protect и нажимаем кнопку Go. В программе я заметил небольшой сбой (версия 1.11c): после защиты оказалось, что она перезаписала и исходный файл. В общем, перед защитой сделай резервную копию исходного файла.

Надеюсь, что этой статьи на начальном этапе будет достаточно для защиты твоих программ.



Серийные номера



Параметры "триальной" версии

Константин Клягин (http://thekonst.net)

СВОБОДУ СОФТУ

О СВОБОДНОМ СОФТЕ ДОСТУПНО И БЕЗ ФАНАТИЗМА ОТ АВТОРА НЕСКОЛЬКИХ ПОПУЛЯРНЫХ OPENSOURCE-ПРОЕКТОВ

Люди с древних времен размышляли о свободе. Философы перевели тонны папируса и бумаги, выводя и формализуя это понятие. Многие квадратные километры кафеля общественных уборных зафиксировали наблюдения простых смертных на эту тему - в чем заключается свобода? Свободе ставили статуи и посвящали оды. Свободная любовь, свободная пресса или же свобода писать на стенах и дверях (или на двери и на стене) лифтов в наших подъездах? Свобода создавать или свобода потреблять? Проецируя все эти вопросы на рынок софта, попробуем разобраться в отдельном сегменте - в свободном ПО.

Оглядимся вокруг. Казалось бы, повсюду мы имеем коммерческие программы: Windows XP и Microsoft Office, игрушки от Id Software, куча небольших shareware-утилит. И если не учитывать специфику отечественного рынка, за все нужно платить. Неутешительная мысль, если учитывать среднюю толщину кошелька граждан. А в случае если компьютер с такими продуктами стоит не дома, а где-нибудь на фирме или в интернет-клубе, дело может закончиться совсем плохо. Вполне вероятными гостями могут быть представители правоохранительных органов, вместо опохмела решившие всем отделением пойти поохотиться на "пользователей пиратского софта".

На компьютерах большинства секретарш, бухгалтеров, банкиров, дантистов и бандитов действительно изо дня в день работает коммерческий софт. Такова общая ситуация в области пользовательских систем. Однако если мы выйдем в интернет и обозрим его просторы любым доступным браузером, то обнаружим совсем другую картину. Многие web-сайты работают на сервере Apache под Linux, а в строке URL зачастую приходится видеть расширение скриптов .php, которые иногда выдают ошибки базы данных MySQL. Все это: Linux, PHP и MySQL, - уже из другого мира, из мира свободного программного обеспечения и открытых исходных текстов.

ПОЛИТИКА

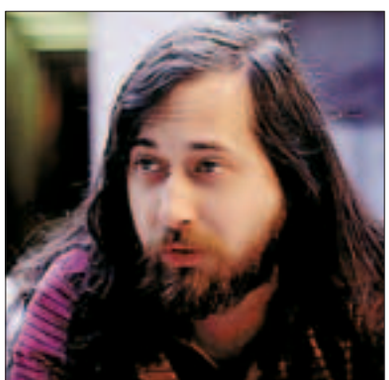
Итак, у нас есть несколько примеров свободного софта. Но пока непонятно, по каким критериям можно считать программу свободной. Скажем, все ясно с заключенным, который только что перестал быть им - он теперь "на свободе". "Свободный полет" - это когда тебя никто не держит и у тебя хороший двигатель, а иначе это "свободное падение". Есть еще "свободные выборы" - это уже с политическим душком. В ситуации с определением свободы ПО от политики нам тоже никуда не уйти.

Без сомнения, самый главный политик и идеолог движения за свободный софт - это Ричард Столлман (Richard Matthew Stallman или RMS), основавший в 1984 году проект GNU. У него длинные волосы, борода и безумный взгляд. Ричард много колесит по миру, рассказывая на всевозможных конференциях о пользе свободного ПО и о вреде патентов, о том, как негативно они сказываются на технологии, техническом прогрессе и на потенции.

Программы, в понимании Столлмана, сродни кулинарным рецептам. Действительно, рецепты напоминают софт, с тем лишь исключением, что компилируются они поваром, а не компилятором. Столлман говорит, что никто не сочтет это преступлением, если вы вдруг поделитесь рецептом со своим соседом, дабы тот смог приготовить блюдо на свой вкус. Кроме этого, всегда хорошо знать, чем именно тебя кормят: какие ингредиенты этого блюда и в каких пропорциях их использовали.

В кругах особенно страстных фанатов Столлмана называют архангелом, пророком и восьмируким Буддой. Я воздержусь от традиционных похвал и рукоплесканий. На мой взгляд, мужчина немного не в себе. Программеры по своей сути аполитичны, и им, как правило, фиолетово, под какой именно лицензией распространять собственное детище. Также, как правило, они не против, если их исходником воспользуется кто-то еще. Как это сформулировано и преподнесено - дело десятое, лишь бы код был хорошим.

Но Столлман гораздо более придирчив к формулировкам и названиям. В частности, в них заключается причина конфликта Столлмана с создателем свободной операционной системы Linux - с Линусом Торвальдсом (Linus Torvalds). Хотя весь исходник ее ядра и так распространяется под свободной лицензией, Линуса он упрекает в том, что тот называет свою операционку Linux, а не GNU/Linux. Столлман настаивает на том, что префикс GNU обязателен, потому что Линус написал только ядро, в то время как базовые утилиты, как и много других полезных программ под Linux, написаны именно в рамках проекта GNU.



Ричард Столлман aka RMS

Суть проблемы в том, что большинство пользователей не понимают, что Linux - это ядро, а не весь набор программ. Они могут подумать, что весь их дистрибутив создан Линусом без участия Ричарда, что и пугает последнего. Мало того, что Торвальдс однажды согласился выпустить свой собственный код под GNU-лицензией, теперь он должен еще и переименовать свой продукт.

Несмотря на все это, назвать Столлмана бездельником и пустомелей тоже нельзя. Именно ему принадлежит

Эрик Рэймонд aka ESR

Линус Торвальдс - автор Linux

Логотип GNU

авторство таких вещей, как редактор Emacs, отладчик GDB и, самое масштабное, GCC (GNU Compiler Collection - набор компиляторов). Осмелюсь предположить, что согласно его логике, дабы неискушенные в политике пользователи не посчитали его только оратором и писателем, Ричарда стоит переименовать в Emacs/Stallman или Stallman/GCC :).

ЧЕГО ХОЧЕТ FSF?

Во второй половине 80-х была основана организация Free Software Foundation, которая с тех пор борется за свободу ПО. Главное оружие FSF - лицензия GPL (GNU Public License), обеспечивающая правовую базу принципу распространения свободных программ. GPL - лицензия, совместимая с концепцией copyleft (название - издевательство над словом copyright). Copyleft оговаривает основные признаки, по которым программа считается свободной. Такая свобода имеет несколько уровней:

- уровень 0: свобода запускать программу с любой целью;
- уровень 1: свобода изучать и изменять программу;
- уровень 2: свобода делать копии программы для третьих лиц из дружеских соображений;
- уровень 3: свобода улучшать программу и публиковать такие улучшения во имя выгоды сообщества.

Очевидно, что уровни 1 и 3 требуют доступа к исходному коду.

Так как главным врагом прогресса в сфере ПО FSF считает патенты, борьба с ними ведется следующим образом. Согласно лицензии GPL, обладателем патента на исходный код программы является FSF. Поэтому в случае нарушения лицензии, например, при использовании исходника в коммерческом проекте с закрытым исходным кодом, FSF от имени всей организации будет оспаривать авторское право.

ОППОЗИЦИЯ

Сформулировав идеи свободы ПО и расписав ее уровни, Ричард Столлман продолжил бурную пропаганду своих идей. Когда в 1991 году FSF обрела в своем активе ОС Linux, среди ее разработчиков, помимо Линуса Торвальдса, оказался американец Эрик Рэймонд (Eric Steven Raymond). Окончательно устав от идеологии FSF и риторики Столлмана, в 1998 году он вместе с Джоном "Бешеной Собакой" Холлом (John maddog Hall), Ларри Августином (Larry Augustin) и еще несколькими товарищами организовал альтернативную организацию - Open Source Initiative, или, сокращенно, OSI.

Основанная на идее исключительно технического, а не идеологического превосходства программ с открытыми исходными текстами, OSI стала вторым сообществом программеров, продвигающим идеи открытой разработки ПО для масс.

Конечно, как и у всякого движения, у Open Source есть свой "Капитал", "Малая земля" и "Рухнама" в одной обложке. Принадлежащая перу Рэймонда книга называется "Собор и базар". Под этими двумя терминами понимаются два разных подхода к разработке софта.

- Собор. В рамках этой модели исходники открываются только когда выходит новая версия. В остальное время разработка и тестирование ведутся узким кругом разработчиков. В качестве примера Эрик в том числе приводит и проекты Столлмана GNU Emacs и GCC. Стоит ли говорить, что разработка коммерческих проектов ведется похожим образом, с тем лишь исключением, что совершенно невозможно получить доступ к исходных кодам.

Эмблема OSI

- Базар. Модель, в рамках которой код всегда доступен публике. Она может смотреть его, ковырять и указывать на баги и проблемы еще не вышедшей версии. "Когда глаз много, любая ошибка обнаруживается быстро" - девиз, придуманный для этого подхода Рэймондом.

Базар. Давка. Крики. Торговля. Лужи воды с запахом селедки и молочных продуктов. Именно так Эрик и советует разрабатывать софт. Действительно, когда в разработке может участвовать любой желающий, софт к релизу становится намного стабильнее, чем он был бы при "соборной" разработке. Сказывается обильное тестирование. К примеру, именно так по сей день ведется разработка ядра Linux. А самого Линуса вместе с Ларри Уоллом (Larry Wall, автор языка Perl) и Гуидо ван Россумом (Guido van Rossum, отец Питона (Python) можно найти в списке участников OSI.

ПИСАТЬ ИЛИ НЕ ПИСАТЬ?

На первый взгляд, все opensource-разработчики могут показаться альтруистами, чистыми невинными ангелами, делающими большую работу на благо общества. Но задумайся, возможно ли, что кто-то станет тратить сотни часов собственного свободного времени исключительно чтобы сделать жизнь других комфортнее и при этом не потребует абсолютно ничего взамен?

Так как исследованиями opensource уже относительно давно занимаются разные серьезные организации, для того чтобы узнать мотивацию таких товарищей, можно взглянуть на результаты опросов. Например, на исследование, проведенное в 2002 по заказу Европейской Комиссии университетом в городе Маастрихт, Нидерланды. Согласно финальном отчету, основные причины, побудившие разработчиков заниматься opensource, распределились следующим образом (в порядке убывания популярности):

- научиться чему-то новому или развить какие-то навыки;
- поделиться знаниями с другими;
- поучаствовать в новом способе коллективной разработки;
- улучшить существующие opensource-продукты;
- потусоваться на opensource-сцене;
- доказать, что софт не должен быть чьей-то собственностью;
- решить проблему, которая не могла быть решена с помощью коммерческого софта;
- получить дополнительные преимущества, которые можно было бы использовать при приеме на работу;
- получить помощь в реализации своих идей для ПО.

К этому стоит добавить, что авторство популярного проекта представляет собой еще и политический капитал - возможность сформировать ре- »

www.fsf.org - сайт Free Software Foundation.

www.gnu.org - сайт GNU.

www.opensource.org - сайт Open Source Initiative.

путацию, которую при желании всегда просто преобразовать и в денежную выгоду, то есть получить хороший заказ или устроиться на работу, связанную с тематикой такого проекта.

Конечно, браться стоит за что-то такое, чем ты сам станешь пользоваться. В моей практике неоднократно бывало, что какая-то уже существующая opensource- или коммерческая программа не устраивала меня по тем или иным причинам, и именно это служило поводом к созданию нового продукта. В конце концов, нет смысла изобретать колесо без повода для этого.

ПЛЮСЫ

■ По своему опыту разработки opensource-продуктов скажу, что к плюсам относится имя, некоторая известность в компьютерных кругах, а также возможность позаниматься чем-то интересным. Причем если у проекта толковая идея, у него обязательно заведутся пользователи. Направление, в котором автор ведет собственный проект, зависит только от него самого.

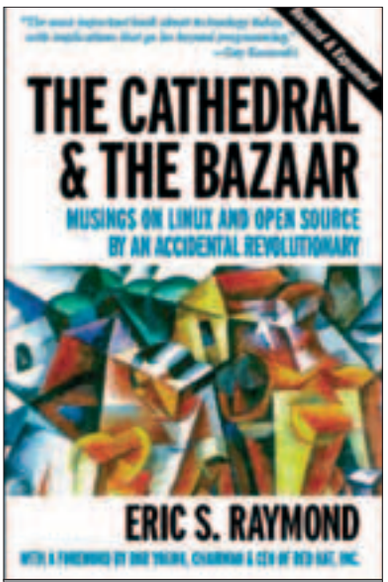
А вот желание заработать денег на простом opensource едва ли оправдается. Многие проекты предоставляют возможность пользователям отблагодарить автора за его труд - так называемые donations (пожертвования). Как правило, donations выглядят так: на сайте программы вывешивается текст вроде "если вам нравится программа и вы хотите отблагодарить за нее автора, кликните здесь". Ну и перевод по webmoney, paypal или с карточки. Так вот, вывесить такое у себя можно, но возлагать на такие пожертвования больших надежд не стоит. Что-то будет капать, но очень редко и помалу. Удвоения ВВП, то есть зарплаты, явно не получится. Такова человеческая натура: если платить не обязательно, то, скорее всего, никто и не заплатит.

Один из принципов свободного ПО - "works fine for me" (у меня работает нормально, меня устраивает), поэтому еще один плюс состоит в том, что ты ничем никому не будешь обязан. Хочешь - пишешь, не хочешь - не пишешь. А можно и забить совсем, и никто не потребует за это компенсации в соответствии с действующим законодательством и согласно условиям контракта.

В любом случае, у тебя будет полностью подконтрольная прикладная программа, которая будет устраивать тебя на 100%. А ежедневно пользоваться собственноручно написанным софтом, скажу тебе, - особенное ощущение.

МИНУСЫ

■ Очевидно, что ни один опрос общественного мнения на тему opensource не раскрыл его проблемных сторон. Думаю, что за такие вопросы исследователей вполне могли бы об-



Оригинальная обложка книги Рэймонда о моделях разработки софта

винить в сговоре с MicroSoft, а от особенно рьяных фанов движения вполне можно было ожидать и кучи под дверью. Тем не менее, минусы есть.

Так как возможность материальной выгоды от opensource у нас отпала, осталось моральное удовлетворение и приятное хобби. Вот проект. Он растет, развивается, привлекая новых пользователей. Изредка на сайтах и в журналах появляются рецензии на проект , в которых указывается и имя разработчика (что тоже нелишне, например, при устройстве на новую работу). Пользователи пишут в рассылку или почтой, жалуются на проблемы и предлагают новые функции. Вот здесь с функциями и багфиксами возникает вторая проблема. Оказывается, что большинство просто хотят чего-то даже не задумываясь о том, что исходный код программы доступен свободно и что так предоставляется возможность самостоятельно сделать нужное изменение и прислать его в форме патча. Конечно, если автор энтузиаст и, так сказать, на подъеме, то он сделает все сам. Но со временем энтузиазм проходит и поначалу интересный проект переходит большей частью в рутину.

Статуя Свободы

Еще одно наблюдение. Если кого-то тыкнуть носом в исходник и сказать "вперед на патчи", то такой пользователь скорее всего скажет "не умею программировать", "я не знаю языка, на котором написан софт" или "у меня нет времени". Если услышишь такие отмазки раз сто, начнешь изрядно раздражаться. А особенно неприятно, когда твое нежелание учиться пытаются компенсировать критикой кода вроде "нет комментариев" или "плохой код". Говорил всегда - скажу и здесь поговоркой. Нечего зеркало винить, коли рожа крива.

После пользователей, которые горазды только разговаривать, идет вторая категория - тщеславные юнцы. Это люди, которые присылают патчи, состоящие из одной строки добавленного кода и пяти строк с описанием того, кто прислал этот патч. Можно игнорировать.

Ну и, в конце концов, среди в общем благодарной публики появляются откровенно наглые персонажи. То ли они считают, что автор обязан заниматься проектом, то ли думают, что именно они заслуживают благодарности только за то, что являются пользователями, но настроение испортить временами способны. Типичный сценарий:

П: *У меня не работает то-то.*
Р: *rtfm.*
П: *Вот ты какой! Не буду больше программой пользоваться.*

Вот и славненько. Мы дебилов не заказывали. Или вот:

П: *Хочу такую-то функцию.*
Р: *Хочешь - сделай.*
П: *Но ты же автор! А я - пользователь! Я прошу функцию! Ты че?*
Р: *До свидания.*

Opensource = works fine for me. Как ни странно, выходит, что главная отрицательная сторона свободного ПО - пользователи. Вернее, их наименее вменяемая часть.

Здесь мы первый раз идем вразрез с процитированным в начале статьи сравнением программ и рецептов приготовления еды. Ни один вменяемый кулинар не будет требовать от автора рецепта изменить что-то в творении. Он сам сделает все необходимые изменения.

ДРУГИЕ СПОСОБЫ ПОУЧАСТВОВАТЬ

■ Менее напряжным вариантом участия в opensource-разработке яв-

www.sourceforge.net - сайт для коллективной разработки свободного ПО. Каждому проекту бесплатно предоставляется хостинг, CVS, почтовая рассылка и серверы для сборки.

www.savannah.gnu.org - похожий на SourceForge сервис, предоставляется проектом GNU.

freshmeat.net - самый большой каталог свободного софта

ляется содействие разработчику, выражающееся в предоставлении патчей. Напряга уже меньше хотя бы в силу того, что приходится взаимодействовать со свойской публикой - с разработчиками. Особенно актуально, если продукт ориентирован на конечного пользователя, проявляющего себя во всем вышеописанном разнообразии. Пожалуй, лучший способ помочь себе и другим - улучшить или исправить понравившийся проект прежде всего для себя.

С точки зрения разработчика программы, использующей какую-то общедоступную библиотеку, такой вид взаимодействия является идеальным. В этом случае действительно выигрывает прогресс и все участники процесса разработки, каждый из которых получает благодаря себе и своим соратникам постоянно совершенствующийся инструмент. Здесь примером для меня всегда была кроссплатформенная библиотека libcurl, реализующая различные сетевые протоколы. Чтобы не ограничивать ее использование, авторы выпустили libcurl под очень свободной opensource-совместимой MIT/X лицензией. Дело в том, что GPL требует, чтобы программы, использующие какой-либо лицензированный ею код, также распространялись под GPL, а значит, с открытыми исходниками. Очевидно, что такой подход делает невозможным использование библиотеки в коммерческом продукте. Кстати, можно подумать и о таком варианте, если есть желание написать библиотеку.

Впрочем, если просто хочется встрять в разработку чего-нибудь, можно сходить на страницу Help wanted сайта sourceforge - http://sourceforge.net/people, почитать объявления и кинуться в омут с головой.

sourceforge.net - сайт для коллективной разработки ПО

Savannah.GNU.Org - заповедник антилоп

Если проект запускается для того чтобы вместе с тобой его разрабатывали другие, о нем можно забыть.

www.freshmeat.org - свежее мясо, самый большой каталог по большей части свободного софта под Linux.

www.infonomics.nl/FLOSS - результаты исследования свободного ПО по заказу Европейской Комиссии.

КОЛЛЕКТИВНАЯ РАЗРАБОТКА ИЛИ СВАЛКА ПРОЕКТОВ?

Когда в 1997 году Linux-энтузиаст Патрик Ленц (Patrick Lenz) из Германии создавал сайт freshmeat.net. Он, наверное, и не подозревал, что его детище станет крупнейшей коллекцией open-source-программ в Сети. И уж тем более в его голову наверняка не приходило осознание того, что часть такой коллекции обречена стать свалкой давно никем не поддерживаемых opensource-продуктов. И хотя сайт имеет механизм автоматической борьбы с замусориванием в виде проверки ссылок на файлы дистрибутивов и на домашние страницы проектов, на freshmeat.net достаточно программ, не обновлявшихся по три-четыре года.

Почему же так происходит? Отчего так часто авторы забрасывают свои детища? Отсутствие мотивации продолжать проект, отсутствие ожидаемых результатов: пользователь не пошел (крокодил не ловится, не растет кокос) или никто не присоединился к затее. Здесь хочется прояснить еще одну деталь. Если проект запускается для того чтобы вместе с тобой его разрабатывали другие, о нем можно забыть. Во-первых, твоя идея не нужна никому, кроме тебя, поэтому держаться, скорее всего, она будет на твоем и только твоем энтузиазме. Может быть, потом, когда программа будет готова и если ей воспользуется еще кто-то, подтянутся еще один-два человека.

Что касается мотивации, то она будет сильной настолько, насколько ты сам будешь пользоваться результатами своей работы. Выживает только то, что автор пишет для себя и что он сам использует по мере необходимости. Если ты возьмешься за сервер баз данных, но при этом продолжишь пользоваться MySQL, то ожидать от других того, что они MySQL'ю предпочтут твою поделку, как минимум наивно.

СВОБОДНОЕ ПРОТИВ КОММЕРЧЕСКОГО?

Мир свободного ПО интересен и разнообразен. Успешно развиваются аналоги программ, которые в недавнем прошлом были доступны только на коммерческой основе. Adobe Photoshop можно заменить на GIMP, IIS на Apache, Windows на Linux и так далее. Бесплатно. Почему же рынок коммерческого ПО еще жив? Все ли можно заменить на opensource?

Возвратимся к аналогии с кулинарными рецептами. Представим, что в мире кулинарии нормой стал свободный обмен рецептами и что именно в таком мире мы пошли в культпоход в ресторан. Выпив, закусив и отведав горячего, попросим у официанта вместе со счетом рецепт заказанных яств. Едва ли посчастливится получить что-то кроме счета, особенно если ресторан - не простая забегаловка и если кушали не чизбургер, а что-нибудь поизысканнее. В серьезных ресторанах способ приготовления фирменных блюд держится в строжайшем секрете. Если рецепт популярного блюда попадет к конкурентам, такая утечка информации приведет к потере клиентуры.

Видимо, вера в свободное ПО подразумевает и веру в свободу кулинарии. Но как доказал поход в ресторан, это не совсем так. Впрочем, никто не отменял и "Книгу о вкусной и здоровой пище", рецептами из которой действительно можно делиться. С программным обеспечением дело обстоит примерно так же.

Противостояние между свободным и коммерческим ПО многие стремятся свести к вражде между FSF и Microsoft. FSF кажется многочисленной гвардией борцов за правое дело, а MS - ненавистной всем империей зла. Это тоже не совсем верно, так как на утверждении, что за копирование и использование софта нужно платить, вместе со злыми MS зарабатывают на жизнь многие тысячи небольших фирм, а также огромное количество одиночек разработчиков shareware-программ. Без копирайта мы не увидели бы многих полезных программ, потому что у всей этой гвардии просто-напросто не было бы мотивов писать их.

Наконец, посмотрим на большие корпорации, ныне широко именуемые друзьями opensource, - Sun, IBM, Oracle и Apple. Все GNU-сообщество кланяется в ножки гяде Эллисону (хозяину Oracle Corporation), который, аки Бог неба Зевс, мечет молнии в сторону Microsoft. Да, он инвестировал что-то в развитие Linux, но его собственные продукты от этого не стали opensource. База данных от Oracle по-прежнему стоит немалых денег. IBM тоже поучаствовала в улучшении Linux. Однако за лицензию на IBM DB2 под Linux придется выложить $337, а Sun Java Webserver стоит полторы тысячи. Продается хорошо.

Едва ли можно построить серьезный software-бизнес лишь на разработке свободного ПО. Раздавать бесплатно, а продавать лишь поддержку, - идея хорошая, но не очень выгодная. Ситуация такова, что компании пользуются opensource-продуктами из-за выгоды, которую те несут будучи бесплатными и обладая всей функциональность своих коммерческих аналогов.

Фирмы вкладывают средства в развитие opensource лишь тогда, когда речь идет о добавлении необходимых им функций, а не из-за идеологии. Просто это дешевле написания собственного софта с нуля. Обычно речь идет о платформах и средствах разработки, которые действительно стоит разрабатывать совместно. Но сложное ПО, направленное на решение конкретной задачи, скорее всего, будет коммерческим. Поэтому место перед монитором найдется для всех: и для opensource-программеров, и для разработчиков коммерческого софта.

Хакер Спец SMS СЕРВИС

РАСШИФРОВКА ТЕРМИНОВ
КАРТИНКИ ДЛЯ МОБИЛЬНОГО
АНОНС СЛЕДУЮЩЕГО НОМЕРА
ОТВЕТЫ НА ВОПРОСЫ

ВСЕ ЭТО ТЕПЕРЬ ДОСТУПНО В РЕАЛЬНОМ ВРЕМЕНИ С ТВОЕГО МОБИЛЬНОГО ТЕЛЕФОНА!

Хочешь узнать ответ на вопрос?

Хочешь узнать о нас больше? Присылай свои вопросы на адрес andrusha@real.xakep.ru

Как стать автором статей в журнал ХакерСпец? (код w0031)
О чем мечтает AvaLANche? (код w0122)
Как проводит свободное время Dr.Klouniz? (код w0123)
Почему SkyWriter часто хихикает? (код w0124)
Какой ноутбук у Gorlum'a? (код w0125)
Какой ноутбук у AvaLANche'a? (код w0126)
На кого учится Dr.Klouniz? (код w0127)

У каждого вопроса есть свой уникальный код (к примеру w0127), который надо послать на короткий номер **4445**. Ответ придет в виде СМСки.

Хочешь узнать, что будет в следующем номере "Хакера Спец"?

Для получения анонса, что будет в следующем номере "Хакера Спец", отправь "ON NS" (без кавычек!) на короткий номер **4446**. Анонс придет в виде СМСки. Но не сразу, а за 3-5 дней до выхода журнала. Теперь ты не проспишь новый номер и будешь раньше всех знать, еще не купив журнал, о чем он будет.

Хочешь узнать, что значит термин?

степпинг	(код w0088)	кодировка	(код w0030)
интерфейс	(код w0010)	тестер	(код w0103)
утилита	(код w0017)	дамп	(код w0104)
эмулятор	(код w0044)	дескриптор	(код w0003)
брандмауэр	(код w0018)	стек	(код w0105)
трафик	(код w0089)	буфер	(код w0006)
скрипт	(код w0009)	исключение	(код w0106)
протокол	(код w0076)	мидлет	(код w0107)
дизассемблирование	(код w0090)	обфускатор	(код w0108)
микроконтроллер	(код w0091)	дистрибутив	(код w0016)
архитектура	(код w0014)	документация	(код w0109)
драйвер	(код w0001)	поток	(код w0110)
компилятор	(код w0002)	хэширование	(код w0111)
транслятор	(код w0092)	кроссплатформенность	(код w0112)
библиотека	(код w0012)	хост	(код w0019)
ядро	(код w0058)	сокет	(код w0007)
верификатор	(код w0093)	тег	(код w0027)
отладчик	(код w0043)	парсер	(код w0028)
кодек	(код w0066)	браузер	(код w0113)
спам	(код w0094)	инсталятор	(код w0114)
офшор	(код w0095)	реестр	(код w0115)
крякер	(код w0096)	аккаунт	(код w0116)
бета	(код w0097)	домен	(код w0117)
скин	(код w0098)	девелопер	(код w0118)
сертификация	(код w0099)	флуд	(код w0119)
аутсорсинг	(код w0100)	пиктограмма	(код w0120)
трассировка	(код w0015)	архиватор	(код w0121)
хостинг	(код w0023)	окружение	(код w0080)
баннер	(код w0101)	визуализация	(код w0038)
локализация	(код w0102)	конвертер	(код w0060)

У каждого термина есть свой уникальный код (к примеру w0088), который надо послать на короткий номер **4444**. Расшифровка придет в виде СМСки.

Хочешь фирменный лого на свой сотовый?


1001 1002 1003 1004


1005 1006 1007 1008


1009 1010 5333 6333

Хакер Спец БОНУС!

Его смогут получить только 10 самых активных читателей журнала "Хакер Спец", которые отправят больше всех запросов. Итоги подводятся по итогам месяца.

У каждого логотипа есть свой уникальный код (к примеру 1001), который надо послать на короткий номер **4446**. Ссылка на картинку придет в виде СМСки. Открыв ее, ты скачаешь логотип.

ТЕПЕРЬ В КАЖДОМ НОМЕРЕ...

Подробности: www.i-free.ru, (095) 916-7253, (812) 118-4575, support@i-free.ru. Для заказа картинок включи услугу WAP/GPRS-доступ в Интернет (WAP/GPRS-доступ оплачивается согласно твоему тарифному плану). Проверить возможность закачки можно, зайдя на wap-сайт http://4446.ru. В случае ошибки уточни настройки в службе поддержки твоего оператора. Стоимость запроса на номер 4445 - $0.60 без учета налогов, на номер 4444 - $0.30 без учета налогов, на номер 4446 - $0.90 без учета налогов. В случае ошибочного запроса услуга считается оказанной.

Илья Александров (krot31337@nwgsm.ru)

НЕ ШАРОВАРАМИ ЕДИНЫМИ ЖИВ КОДЕР!

АЛЬТЕРНАТИВЫ SHAREWARE

Если спросить любого программиста о том, как заработать денег на своих программах в интернете, то в 90% случаев услышишь в ответ "shareware". Это хороший способ, безусловно, но он отнюдь не единственный. Есть технологии распространения ПО, приносящие не меньше пользы, чем shareware.

ADWARE

AdWare - это схема распространения ПО, при которой программа является бесплатной, но нагло показывает пользователю рекламные баннеры. При этом программа обычно помечается как freeware, что, в общем-то, не совсем верно. По мнению экспертов, это не самый серьезный способ заработать, хотя многие компании его все-таки используют (сразу вспоминаются ICQLite и Opera). Есть ли преимущества у этой технологии, а если есть, то какие?

Итак, такие программы бесплатны, они содержат полный набор функций и не имеют ограничений по времени использования. На первый взгляд, никакой коммерческой выгоды от них не может быть. Но в главном окне программы размещается баннер, за показы рекламы на котором рекламодатель готов платить деньги. Это первый источник дохода. Баннер наверняка надоест пользователю: это лишний трафик да и просто неприятно. На этот случай в AdWare существует возможность избавиться от рекламы купив программу (как shareware). Это уже второй источник дохода. А бесплатность программы будет способствовать ее дальнейшему распространению. Неплохо.

AdWare-программа может участвовать в баннерных сетях так же, как, например, сайт. Достаточно обратится в любую сеть, подробно описать свою программу и, если она сколько-нибудь популярна, получать деньги с кликов пользователей. Правда, надо отдавать себе отчет в том, что если у программы два скачивания в месяц, будет сложно заработать даже на пиво. Чтобы получить прибыль, программа должна быть по-настоящему популярной в широких массах, из-за этого в качестве AdWare распространяются в основном крупные проекты. Для небольших проектов больше подойдет shareware.

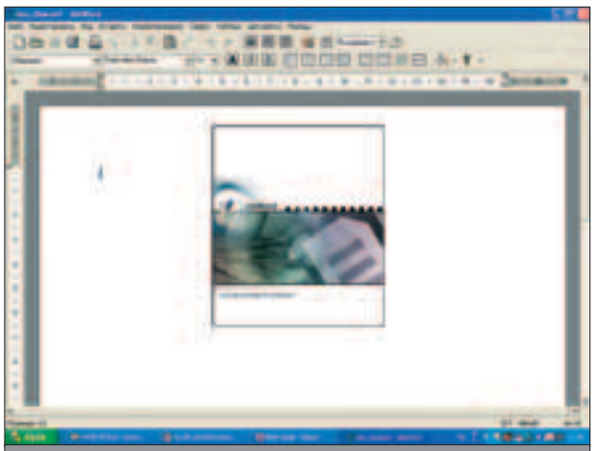
Посмотрим, насколько прибыльным может быть AdWare. Допустим, программой решили воспользоваться 1000 человек. Допустим, каждому из них программа будет показывать десяток баннеров в день, а за один показ платят примерно 0,2 цента. Получается 1000*0,2*10 = 2000 центов = $20 в день. И это практически минимум, показы могут быть более частыми, а пользователей может быть больше (у крупных проектов цифры уже совсем другие). А если еще вспомнить о людях, которые захотят зарегистрироваться, чтобы убрать рекламу... В общем, при определенном желании заработать можно.

Единственной adware-сетью в рунете является Soft.Tbn.ru, работающая на базе баннерной сети TBN.RU и специально созданная для этой схемы. Она позволяет встраивать свои баннеры в любой софт и предоставляет для этого разработчику специальный компонент SoftTBN.dll, который, интегрируясь в ПО, обеспечивает скачивание и показ баннеров. Рекламодатель в этой сети оплачивает только клики, при этом автор софта получает только 50% выплаченных денег. Маловато будет. Поэтому советую обратить внимание на зарубежные сети, в которых, возможно, процент более демократичный: www.radiate.com, www.cydoor.com, www.everad.com.

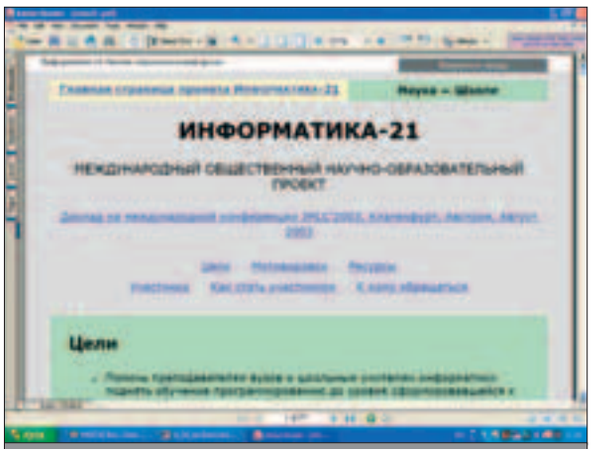
> AdWare-программа может участвовать в баннерных сетях так же, как, например, сайт.

Один из самых популярных AdWare-проектов: серфер Opera

Теперь о грустном. AdWare - развитая ветвь софта, от которой на сегодняшний день отказались почти все (кроме нескольких монстров). Почему? Причин много, но, безусловно, основная - это опасность потери доверия пользователей, которые в наше тяжелое время не склонны выбирать программы, использующие интернет во-



Бесплатный кроссплатформенный текстовый редактор AbiWord



Вот так надо делать Freeware! Adobe Acrobat Reader

преки воле пользователей. Кто мешает разработчику добавить к баннеру троянский код, скажем, ворующий пароли? AdWare в последнее время все больше ассоциируется со SpyWare (программами-шпионами, троянами, кейлоггерами etc). Даже выпускаются утилиты, обнаруживающие и уничтожающие AdWare (Ad-aware, к примеру). Поэтому, прежде чем заняться распространением своего софта по этой схеме, нужно не один раз подумать о том, нужно ли это тебе.

DEMOWARE

■ Довольно популярный способ распространения софта, при котором часть функций программы, в том числе очень важных, недоступна до совершения покупки. Другими словами, пользователь бесплатно скачивает программу, пользуется ей, а когда обнаруживает отсутствие самых необходимых функций, оказывается вынужден купить ее (или выкинуть программу куда подальше и найти freeware-аналог). Лучше всего выпускать в demoware мощные и многофункциональные программы, в которых функциональное ограничение будет заметно, но при этом не будет вылезать на передний план. Что вырезать из демо, а что оставить? Удалишь важное - пользователь не оценит всей пользы программы, незначительное - и человек просто не станет покупать программу, так как все нужные ему функции и так есть. Оптимальный вариант - это когда пользователь видит результат работы, но воспользоваться им не может. Допустим, ты автор наворочennого текстового редактора. В нем можно как угодно круто редактировать и оформлять текст, но функция сохранения будет доступна лишь в полной версии. Если пользователю потребуется сохранить наработанное, он обязательно купит программу ;).

Способ DemoWare особенно хорош в плане защиты от крякеров, так как части кода просто нет, а дописывать его вряд кто-то решится. Crack под demo-программу пользователь вряд ли найдет.

FREEWARE

■ Итак, freeware - это софт, распространяемый бесплатно без каких-либо ограничений. Но если ты думаешь, что от бесплатного ПО нельзя извлечь выгоды, ты ошибаешься! Вот наиболее очевидные задачи, решению которых (в том числе нескольких одновременно) может способствовать распространение freeware: реклама торговой марки и\или имени разработчика, "продвижение" новой технологии или формата (например, Adobe раскрутила формат PDF благодаря Acrobat Reader), реклама более сложной и функциональной коммерческой программы. Все наиболее популярные freeware-проекты тебе, конечно, известны: Mozilla, WinAMP, Adobe Acrobat Reader.

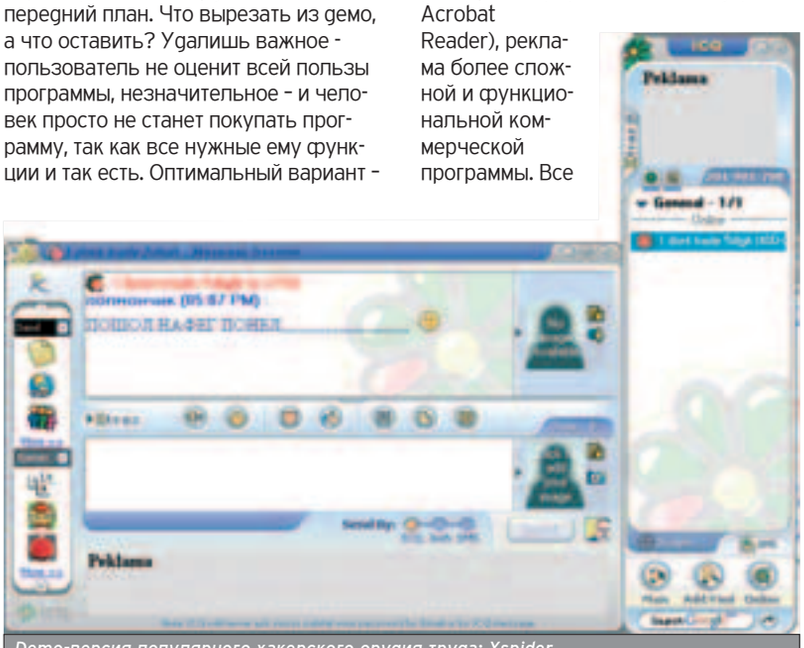
DONATIONWARE

■ Donationware, или "кто сколько может". Никаких ограничений, только надежда на "благодарного пользователя", который заплатит, если программа понравится ему. Понятно, что авторы программ, распространяющихся в виде DonationWare, не рассчитывают на большую коммерческую выгоду. Но если в About соврать, что все деньги пойдут на благотворительные цели и если программа будет не самая плохая, какая-то прибыль, может быть, и появится. Серьезного бизнеса на такой технологии не построишь. Хотя, может быть, ты создашь калькулятор, его увидит Билл Гейтс, твое детище будет использовано в следующих версиях Windows, а тебе на счет переведут пару миллионов долларов. Но будем смотреть правде в глаза - вряд ли такое произойдет ;).

З.Ы.

■ Если после прочтения моей статьи ты будешь распространять программы способами, альтернативными shareware, будем считать, что я выполнил поставленную передо мной задачу. В любом случае ты понял, что shareware - это не единственный способ нарубить капусты в интернете на своих программах.

Не видел программ, подходящих для встраивания рекламы лучше, чем различные мессенджеры. Общаясь, человек не сможет не смотреть на баннеры. Так что давай начинай кодить убойный ICQ-клиент!

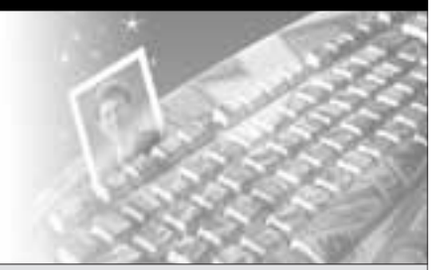


Demo-версия популярного хакерского орудия труда: Xspider

Игра "нашей" тематики распространяется как DonationWare

Степан Ильин aka Step (step@real.xakep.ru)

ЧТО ТАКОЕ COPYRIGHT?



ОСНОВЫ ЮРИДИЧЕСКОЙ ЗАЩИТЫ ТВОЕЙ ПРОГРАММЫ

И спокон веков жизнь в нашем мире омрачалась активностью самых разных мошенников. Теперь можно смело сказать, что они добрались и до интернета. А значит, нужно решать вопрос о том, как защитить свою продукцию от хищных аферистов.

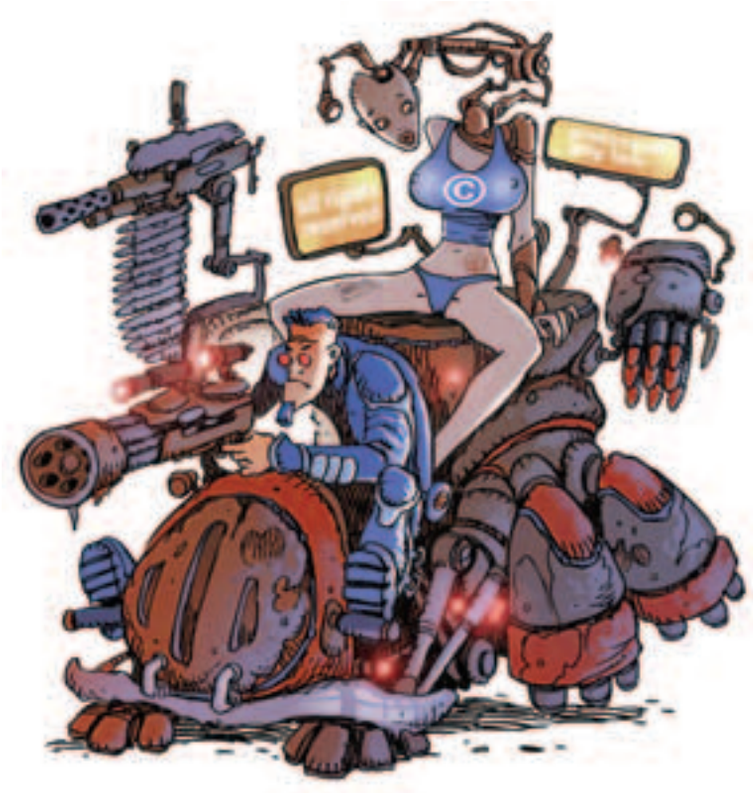
ЧТО ТАКОЕ ЛИЦЕНЗИЯ?

Для начала разберемся с понятием "лицензия" в плоскости программного обеспечения. В это понятие входит не только узаконенное право заниматься определенным видом деятельности, в нашем случае - программированием. В нашем случае уместно другое определение. Лицензия - это разрешение, которые выдается обладателем исключительных прав на объект интеллектуальной собственности (компьютерной программы) и его использование. Правомерность таких лицензий регулируется законами о защите интеллектуальной собственности. В рамках этой статьи я буду ссылаться на законы РФ "Об авторском праве и смежных правах", "О правовой охране программ для ЭВМ и баз данных", "О товарных знаках, знаках обслуживания и наименованиях мест происхождения товаров". В некоторых случаях уместно будет упомянуть и патентные законы РФ, но патентование программного обеспечения пока еще не получило должного распространения. Хотя здесь, как и везде, есть некоторые разногласия. В частности, США из-за давления крупных монополий всеми руками и ногами ЗА введение подобного в практику, в то время как в Европе этому противятся.

Распространение программных продуктов обычно сопровождается заключением двусторонних договоров - лицензионных соглашений. Правообладатель исключительных прав на программу может передать другому лицу (пользователю или, в случае продажи, покупателю) право на ее использование на определяемых сторонами основаниях. Разумеется, эти основания во многом зависят от определенной ситуации.

КАКИЕ ОНИ БЫВАЮТ?

Лицензии, естественно, бывают разные. Опишу типы лицензий и соответствующие ситуации.



В твоей жизни случилось радостное событие: для программы, написанной тобой, нашелся покупатель. Причем покупатель не просто хочет приобрести ПО в целях его использования, но и получить исключительное право на его дальнейшую разработку и распространение. Не вопрос! В этом случае должен быть подписан договор, подразумевающий передачу имущественных прав и исключительных привилегий в отношении программы. При этом ты, как продавец, скорее всего, эти права потеряешь.

Другая ситуация и другой тип соглашения: ты решил поддерживать и распространять разработанную программу самостоятельно. В этом случае уместно передавать конечному пользователю лишь жестко ограниченные права на программный продукт. В большинстве случаев можно ограничиться правом на использование и на техническую поддержку. Другими словами, исключительные права не передаются конечному пользователю, а значит, правообладатель не лишается возможности использовать свою разработку как ему будет угодно.

Выделяют еще один тип лицензий, актуальных в тех случаях, когда программное обеспечение распространяется при условии открытого кода (OpenSource). Пользователь, соглашаясь с такой лицензией, получает не только исполняемые файлы программного средства, но и исходные файлы. Вдобавок к этому часто он получает право на модификацию кода по своему усмотрению, хотя условия дальнейшего распространения переработанных программ строго регламентируются прилагающейся к программе лицензией.

GNU GPL, BSD – НЕ ПРОСТО АББРЕВИАТУРЫ

Существует несколько видов лицензий на бесплатное программное обеспечение с открытым кодом. И в первую очередь к ним относятся генеральная открытая лицензия GNU (GNU General Public License, GNU GPL, или просто GPL), лицензия BSD (Berkley Software Distribution), а также их многочисленные разновидности.

GPL получила наибольшее распространение и на сегодняшний день наиболее отточена в юридическом плане. В ней оговорено буквально все: от правил распространения программы до рассмотренной ситуации нарушения авторских прав. В то же время GPL является несколько более строгой по сравнению с другой популярной лицензией - BSD. Особенно в отношении переработки исходных кодов и их дальнейшего распространения. Например, лицензия BSD практически не ограничивает пользователя в использовании исходных текстов программы. Единственное требование - сохранение в документации текста лицензии и информации об авторских правах. В свою очередь лицензия GPL, кроме перечисленной информации, подразумевает включение исходных кодов переработанной программе. Другими словами, GPL разрешают любую модификацию и дальнейшее распространение, но лишь при условии, что ты соблюдаешь определенные требования, главное из которых - распространение программы вместе с исходным текстом и лицензией GPL. Такие программные продукты нельзя использовать в коммерческих целях - не положено.

В отличие от GPL, лицензия BSD такую возможность вполне допускает. Обозначенные ею положения предполагают возможность коммерческого распространения продуктов, полученных в результате переработки чужого исходного кода. Так или иначе обе лицензии с одинаковой тщательностью описывают вопросы защиты авторского права, а значит, гарантируют торжество справедливости в отношении первоначального автора. В то же время нельзя исключать различных юридических ловушек, особенно в случае с BSD-программами.

```
This is a sample of a BSD style license.
-----------------------------------------------------------------------
Copyright (c) 2000 <SOMEONE>
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ''AS IS'' AND ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

Шаблон BSD-лицензии

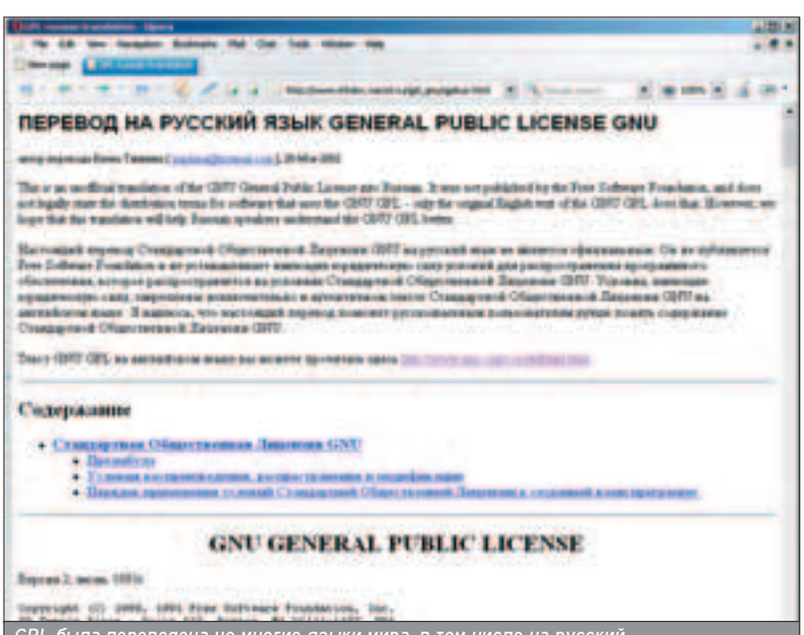
> В отличие от GPL, BSD допускает коммерческое использование программных продуктов.

ОПРЕДЕЛИ ПРАВА

Любую программу, будь она бесплатная (freeware), условно-бесплатная (shareware) или полностью коммерческая, весьма желательно дополнять лицензионным соглашением с конечным пользователем (EULA - End User License Agreement), которое обычно содержит перечень предоставляемых прав, а также обязанностей правообладателя. Условия такого соглашения ты принимаешь практически всегда во время установки какой-либо программы.

Пользователю, как правило, передается минимально возможное количество прав, а разработчик берет на себя минимум обязанностей. Однако в таких документах подробно описываются полномочия пользователя, гарантии разработчика, разъясняется процесс технической поддержки плюс такие лицензии практически всегда запрещают изменение программного кода программы и его непосредственное распространение.

Отличительной чертой распространения свободно софта является условие, согласно которому разработчик освобождается от любых гарантий и обязательств, связанных с функционированием программного обеспече-



GPL была переведена на многие языки мира, в том числе на русский

Условия лицензионного соглашения с конечным пользователем (End User License Agreement) мы принимаем во время установки любой программы!

ния. Любое использование такого ПО пользователь осуществляет на свой страх и риск. Если, к примеру, программа работает с жестким диском компьютера и в результате ее работы потеряются все данные с этого носителя, ответственность будет нести только пользователь, но никак не разработчик.

А где можно взять точную формулировку End User License Agreement? Скажу правду: ее не существует. Однако неплохим образцом для многих разработчиков служит EULA компании Microsoft. Эта корпорация уж точно сделала все, чтобы в случае чего нести как можно меньше ответственности :).

ЗАЩИТИ СЕБЯ САМ

Прежде чем заключать какие-либо лицензионные соглашения, нужно позаботиться о защите своих авторских прав. И на самом деле это не так сложно, как многие думают. Согласно закону "Об авторском праве и смежных правах" авторское право возникает в силу факта создания произведения (литературного, художественного или, в нашем случае, программы ЭВМ) и, в общем, не требует какой-либо регистрации.

Автором программы для ЭВМ или базы данных признается физическое лицо, в результате творческой деятельности которого они были созданы. Однако в случае выполнения трудовых обязанностей, исключительное право на программу будет принадлежать именно работодателю. Если, конечно, специальный договор, заключенный между ними, не предусматривает иного...

Обладатель исключительных авторских прав для оповещения кого-либо о своих правах может использовать знак охраны авторского права, который обычно ставится на каждом экземпляре программы и состоит из символа ©, имени обладателя исключительных авторских прав и указания года создания программы. Статьи 9 и 10 закона "О правовой охране программ для ЭВМ и баз данных" гласят, что автору принадлежат личные (право на имя, обнародование и др.) и исключительные имущественные (право на использование) права на программу. В случае любого их нарушения автор вправе обратиться в суд.

В случае возникновение проблем сложнее всего доказать то, что именно ты являешься автором программного средства. Для гарантированного доказательства такого авторства закон предусматривает возможность регистрации программ для ЭВМ в Федеральной службе по интеллектуальной собственности, патентам и товарным знакам (www.fips.ru). Авторские права в сложных случаях остаются за тем, кому удалось заполучить адвоката получше, поэтому я настоятельно рекомендую тебе в случае чего обратиться в эту инстанцию.

Исключительное право на программное обеспечение может быть в любой момент передано полностью или частично другим физическим или юридическим лицам согласно соответствующему договору. Например, ты решишь полностью продать свой продукт. К такому процессу нужно подходить со всей ответственностью. Если быть совсем откровенным, то рассчитывать только на свои силы здесь не стоит. Грамотный юрист поможет тебе куда больше, чем какая-либо статья закона.

Само название программы было бы неплохо зарегистрировать как товарный знак. Разумеется, при наличии средств и серьезных намерений. Тем более что для работы на международном уровне придется провести эту операцию еще раз, вновь тратить деньги и время. Но зато это избавит тебя от головной боли, например, по поводу возможного перехвата домена имя_програмы.ру, столь лакомого для мошенников.

Copyright.ru - отличное подспорье в изучении правовых вопросов

> Сложнее всего доказать, что именно ты являешься автором программы.

> Грамотный юрист поможет тебе намного больше, чем какая-либо статья закона.

В защите твоих прав может помочь Федеральная служба по интеллектуальной собственности, патентам и товарным знакам

ПАПЕ ЛЕНЬ ИДТИ В МАГАЗИН?

НЕ ГРУСТИ!

РАССКАЖИ ЕМУ ЧТО

в интернет магазине GamePost

* Не нужно выходить из дома, чтобы сделать заказ
* Покупку можно оплатить кредитной картой
* Игру доставят в день заказа

Gran Turismo 4

$59.99

Metal Gear Solid 3: Snake Eater

$65.99

World of Warcraft

$85.99

PlayStation 2
$179.99

*

GameCube
$139.99

*

Xbox
$279.99

Играй просто!

GamePost

Тел.: (095) 928-0360
(095) 928-6089
(095) 928-3574

www.gamepost.ru

GAMEPOST

TiberiuZ (admin@progamer.org.ru)

ИСПЫТАНО НА СЕБЕ

ИСТОРИЯ ОДНОГО ПРОЕКТА...

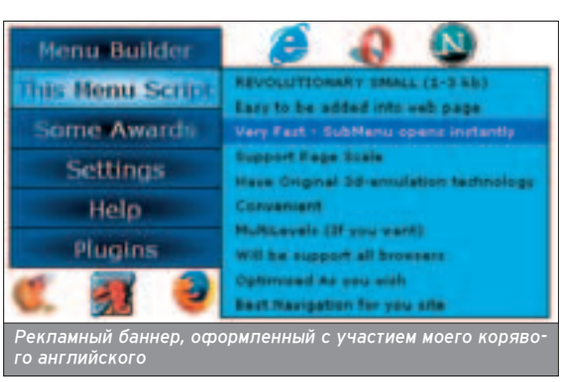
Эта история началась в простом сибирском ВУЗе, на простом факультете на сдаче обычного курсового проекта по программированию. Целью курсового было многоуровневое меню, написанное на любом языке, которое мне захотелось реализовать на DHTML. Все бы ничего, но преподавателю не понравился "мышиный" интерфейс, и курсовой мне пришлось пересдавать много-много раз...

КАК ЭТО БЫЛО...

За долгое время пересдач привередливому преподавателю меню было сильно оптимизировано, и как-то случайно я увидел статью о сравнении десятка лучших аналогичных скриптов. Каково же было мое удивление, когда я увидел, что они жирнее моего меню во многие десятки раз: 30-100 Кб против моих 0,8 Кб. Тогда я и решил продавать этот скрипт. Итак, за пару недель программа была готова, и надо было искать способ продавать ее через интернет. Конечно, я решил посмотреть, как продаются другие программы. Главными требованиями были защищенность и простота регистрации. Под защищенностью подразумевается невозможность запуска программы пользователем без crack'а или keygen'а, под простотой... Как говорится, любая секретарша смогла бы работать в Linux'е. Под наблюдением системного администратора :-).

Есть несколько методов продажи программ, распространяемых через интернет.

- FreeWare - бесплатная программа, все настойчивее насаждающая идеи коммунизма в мире. Поговаривают, что бесплатные программы скоро запретят в США.
- ShareWare или TryBeforeYouBuy - программа работает с ограниченными возможностями до тех пор, пока ее не купят. Есть различные модификации с ограничениями по времени - TrialWare, например, Microsoft XP; по возможностям - CrippleWare, например, XSpider; по удобству - путем постоянно достающих pop-ups, Nagscreen, например, WinRar.
- AdWare - программа работает бесплатно, но при этом что-то усиленно рекламирует, в том числе FlashGet.
- DemoWare - урезанная версия программы бесплатна, но за полную придется заплатить.
- DonatWare - полностью бесплатная программа, автор которой честно просит поделиться деньгами, например, DosBox.
- SpyWare - программа сама бесплатно залезет в чужой компьютер и, наоборот, может потребовать денег за свой уход :).



Рекламный баннер, оформленный с участием моего корявого английского

Для моей программы больше всего подходил вариант CrippleWare, потому что там можно было легко отрубить возможность компиляции скрипта. При этом пользователь мог испытать все возможности и эффекты программы.

Вариантов активации программы также было немало.

❶. Серийный номер. Незарегистрированная версия программы предлагает купить серийный номер. Google предлагает то же самое бесплатно.

❷. HardwareID. Программа предлагает зарегистрироваться с использованием якобы HardwareID, которым является серийный номер раздела винчестера. На языке C это делается командой GetVolumeInformation(). Видимо, авторы этой системы не учли, что потребитель может переставить операционную систему в другой раздел и что ему потребуется получать новый код опять, а то и заново покупать продукт. После пары таких случаев потребитель просто станет игнорировать все предложения о продаже программ этого типа. Более того, у большинства людей, ставящих операционную систему в первый раздел винчестера, серийный номер одинаков, вдобавок есть программы типа CloneHDD, которыми очень часто ставят систему в больших фирмах. Поэтому получается, что HardwareID у разных людей может легко оказаться одинаковым и придется как-то этих людей различать, например, по адресам электронной почты... Опять-таки эта система может быть проломлена поиском серийного номера или покупкой одной копии и ее тиражированием на множество компьютеров. Недостатком этого метода является нежелание человека "светить" свой адрес электронной почты из-за боязни нашествия спама. Даже если у потребителя неограниченное количество денег :-), ему гораздо легче и быстрее найти серийный номер воспользовавшись Google.

Дядя Google с удовольствием подскажет тебе серийные номера ко многим программам :-)

❸. WebMoney SoftWare Activation Service.

Суть в том, что программа привязывается к уникальному номеру клиента WMID.

Недостатком является обязательное присутствие на компьютере покупателя WebMoney Keeper, который к тому же можно скопировать на несколько компьютеров.

❹. Аппаратная привязка [программно].

Программа считывает серийный номер винчестера и проверяет правильность ключа в reg.key. Если ключ верен, программа будет зарегистрирована, а иначе будет доставать пользователя просьбами купить ее. Итак, потребитель решился и ткнул кнопку On-Line Registartion. Программа открывает в браузере список платежных систем, скрыто отсылая серийник винчестера и атрибуты платежа. Web-сервер

Потерять клиента, желающего купить программу, значит потерять реальные деньги.

принимает подтверждение платежа от одной из платежных систем и заносит тот серийник в базу данных. После оплаты программа соединяется с сервером и в случае подтверждения платежа принимает ключ разблокировки, который записывается в reg.key.

Вот что происходит, если смотреть "глазами потребителя":

❶. Кликаем On-Line Registration;

❷. Оплачиваем счет;

❸. Закрываем окошко Thanks for registration.

Этот вариант не поддерживался нормально американскими перепродавцами программ: regsoft, regnow, sharelT, ... К тому же они брали "небольшую" комиссию за продажу программ - 40-80%. Поэтому я решил отличиться и не пользоваться регистраторами, а сделать все самому.

Итак, в виде плана все выглядело просто, но на практике сразу возникло много сложностей. Например, как считать серийный номер жесткого диска? Запрос в Google рассказал, что надо скачать исходный файл программы с winsim.com. Жалко, что он не сказал, как его встроить в программу. Скачанный файл содержал много вкладок, написанных на ассемблере, и выдавал полсотни ошибок при подключении библиотеки vcl.h. В общем, пришлось оставить его в отдельном файле. Естественно, он должен был передавать считанный серийный номер в основную программу, и ничего другого, кроме как писать его в текстовый файл, я не придумал. А чтобы его не подделали, я защитил его md5-подписью. В случае же когда обмен данными не мог состояться, программа писала lib.exe broken. Зачем я так подробно написал? Мне сегодня (через пару месяцев после описанного) пришло письмо следующего содержания: "When I try to register it tells me that lib.exe is broken. Please help. Thanks, Paul Reitzner, Davidson Management Services". Потерять клиента, желающего купить программу, значит потерять реальные деньги. А ведь официальная цена программы $28. Сейчас я реально ощущаю себя по другую сторону баррикады. Теперь я не ищу способ пользоваться платным программным обеспечением бесплатно, а изучаю способы заставить других людей платить за мои продукты. Вот тогда и встал вопрос: как же эти платежи принимать из-за границы? Есть много платежных систем: webmoney.ru, dengi.yandex.ru, rapida.ru, rupay, assistid.ru, paypal.com, e-gold.com, visa.com и др. Для приема платежей из России я выбрал Rupay, так как он принимает платежи 20-ю способами в СНГ, для заграницы подошел E-Gold.

Для приема заграничных платежей отлично подошел E-Gold

КРАТКИЙ ПЛАН ПРОДАЖИ ПО

❶. Пишем программу :-).

❷. Делаем в программе механизм защиты серийным номером, который зависит от установленного аппаратного обеспечения.

❸. Заливаем программу в интернет.

❹. Подключаем механизм продажи или регистратора. Например, Plinus.com

❺. Прописываем программу по всем адресам на hardwaremarket-ingresource.com и swrus.com, с помощью pad_file.xml

❻. Получаем кровные денежки.

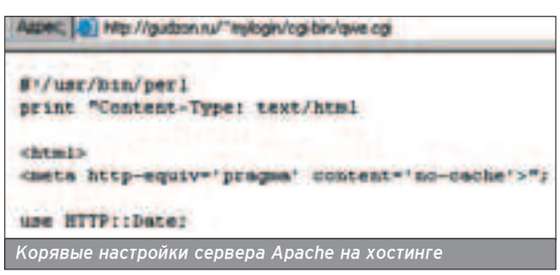
Хостинг за 40 рублей в месяц

"ДАРМОВОЙ" ХОСТИНГ

Короче, я не захотел платить кому-то проценты и решил все сделать сам ;). Тем более что это не казалось таким уж сложным. Во-первых, мне нужен был хостинг. Думаю, не стоит объяснять, почему нельзя размещать сайт своего продукта на narod.ru. Весь интернет завален объявлениями типа "hosting $2,5/month" - на самом деле в эту сумму не включен вступительный взнос и еще десяток платных услуг. "Профессиональный хостинг Агава" - их цены меня тоже не устраивали. Mastak.ru, лучший хостинг на двух континентах, оказался полурабочим. В общем, неизвестно, каким краем меня занесло на gudzonhost.ru, но их тарифы были на порядок ниже. Они предлагали как раз то, что было нужно за 40 рублей в месяц: 100mb/1250mb/php/999mysql/999mails/999domains... Первым делом я купил у соседа 15WMZ=15$. Потом я приобрел месяц хостинга за $1,5 и .com-домен за $9. Также, пока не обновились DNS сервера, мне временно дали адрес для испытания сайта: http://gudzon.ru/~mylogin.

На хостинге была возможность пользоваться бесплатным https через адрес наподобие https://gudzon.ru/~mylogin/.

Каково было же мое удивление, когда я ввел в браузере адрес моего тестового скрипта https://gudzon.ru/~mylogin/cgi-bin/qwe.cgi и сервер не выполнил скрипт, а распечатал его содержимое. А ведь в нем хранилась функция, преобразующая hardwareID в UnlockKey. Другими словами, все шесть временных адресов были открытыми. Причем это не касалось скриптов на php, а ошибка существовала только для cgi-скриптов. »



Корявые настройки сервера Apache на хостинге

"Нехороший" Rupay неожиданно пришлось сменить

Google быстренько отправил меня на WM-форум

Четыре месяца аренды полноценного места на web-сервере обошлись в 15 рублей.

При этом весь мой сайт был написан на perl!

Потом я неделю заваливал службу поддержки письмами с просьбами "исправьте ошибку". Администратор исправлял все ошибки по одной и потом просто снес все временные адреса и бесплатные SSL. За это время я скачал методичку по php и переписал весь сайт на этот язык.

Следующим шагом был автоматический прием платежей. Теоретически все было просто. Надо было заслать в скрипт http://rupay...ndex.php параметры платежа. После этого на мой сайт приходило сообщение с подтверждением платежа: информация о сумме, дате, номере товара и электронной подписи md5. Была одна проблема. Минимальный размер оплаты $5, а у меня оставалось всего 15-9-1,5=4,5$. Пришлось занять один бакс у администратора хостинга ;). Сразу забегу вперед: через пару недель мне предложили переехать на новый быстрый сервер и дали за это три месяца хостинга бесплатно. Короче, получилось четыре месяца аренды полноценного места на web-сервере за 15 рублей :).

МАЛЕНЬКИЕ ХИТРОСТИ

Купив сам у себя программу несколько раз, я отладил систему, казалось бы, идеально... Но вот тут, как всегда, появилось одно злобно-трусливое и очень подлое "но". Rupay сочли прием платежей из-за границы нерентабельным и закрыли его, несмотря на протесты на форуме. Пришлось искать другие пути. Для приема платежей из-за границы я прилепил E-Gold, а для России - WebMoney.

WM также покорился мне не с первого раза: электронная подпись не хотела сходиться. Google отправил меня на WM-форум, в котором эта проблема не одну неделю обсуждалась с местным администратором. Он долго не мог понять, почему у людей на php и perl md5-подпись не сходится, а ссылался на то, что он знает только asp. Оказалось, что в perl и php результатом функции md5 является строка в нижнем регистре, а на ASP - в верхнем. После замены в моем скрипте md5($a) на strtoupper(md5($a)) все заработало. Тот топик популярен и по сей день и находится по адресу google->"site:webmoney.ru md5 не сходится" :-).

Теперь web-сервер мог принимать платежи, но надо было принимать данные от web-сервера программой, написанной на C Builder. Тогда я толком не знал, что такое XML и тем более не знал ничего путного о сетевых протоколах. Так как в моей программе был браузер для просмотра меню, я решил использовать его и для регистрации. Итак, человек жмет кнопку Online registration и:

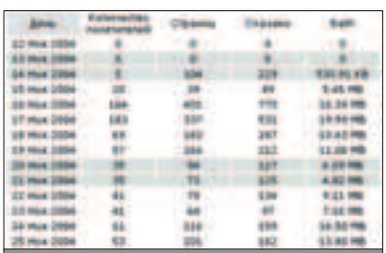
```
Browser->Navigate(WideString("http://my.com/?hid="+GetHID()+"&ver=1.04"),...)
//Передаем web-серверу HardwareID и текущую версию программы.

OnDocumentComplete(){
AnsiString Url=Browser->LocationURL;
if(Url.Pos("about:blank#key=")==NULL)return;
if((UnLockKey=Url.SubString(17,32)).Length()!=32)return;
if(!Validate(UnLockKey))return;
StrToFile("reg.key",SerialKey);
//Получаем код разблокировки программы через тот же url. Готово, программа зарегистрирована. Не знаю, во сколько строк это реализовал бы профессиональный программист, но, думаю, на несколько страниц растянул бы точно.
```

Как написана функция Validate(), рассказывать не буду. Это коммерческая тайна. Но скажу, что код разблокировки в ней не сравнивается, а обрабатывается и только потом сравнивается. Это немного повышает защищенность программы. Также в ней сложно подделать серийный номер, считываемый с диска, так как он не хранится в памяти. Еще я подумывал о том, чтобы добавить к программе проверку crc32, вычитанную на xakep.ru, но забил на это дело - уж больно все сложно получается :-). Все равно, если захотят взломать, взломают даже ради спортивного интереса. Главной цели я добился - для получения программы было три главных пути: покупка, crack, keygen. А большинство пользователей на дух не переносит десятки порно-toolbars, забитые рекламой favorites, и прочие проблемы, напрямую зависящие от количества использованных кряков.

РАСКРУТКА

Теперь все было готово к последнему шагу - к раскрутке. Для того чтобы программу покупали, я добавил ее в download.ru, причем сделал это случайно в пятницу вечером, даже не за-

Три дня - и программа заняла первое место на download.ru :-)


Все исходные файлы ты найдешь на диске журнала и на download.net.ru

думываясь над этим фактом. Прикол был в том, что за три дня программа забралась аж на первое место и за это время ее успели скачать несколько сотен человек.

Но все равно мне были нужны богатые западные потребители. Google посоветовал мне заглянуть на software-marketingsource.com, так как там был список из 400 сайтов, распространяющих ПО. Добавлять программу в такое количество сайтов было, мягко говоря, лень. Тут мне помогла технология PAD, суть которой проста, как алгоритм пузырька. Специальный редактор сохранил в pad_file.xml все данные о моей программе: описание, url's, скрины, информацию о совместимости и т.д. Потом этот файл заливался на web-сервер. А далее, чтобы не писать 400 раз одно и то же везде, просто указывался url этого файла. Эту технологию поддерживало только 150 сайтов, с ними я управился за три часа. Остальные сайты я просто проигнорировал. В первые дни программу скачивали 50 раз в день, а потом это количество снизилось до 30-ти. В общем, за первые полмесяца программу скачали 544 человека.

Как оказалось, Google дает небольшой поток посетителей. До 60-ти в месяц при моем словосочетании. Впрочем, может, все дело в неправильной оптимизации сайта под поисковые системы. Очень хороший способ - упомянуть на сайте названия программ конкурентов (например, в тестировании) и слова crack, serial. Половина трафика из поисковиков получается именно таким способом, причем приходить будут люди, заинтересованные в использовании аналогичной программы.

Есть еще один вариант увеличить приток с Google - заказать дополнительную рекламу в нем же самом. Это называется AdWords. В этом случае Google показывает объявление посетителям, и за каждого человека, прошедшего по ссылке, придется отстегнуть от пяти центов. Мои конкуренты активно пользуются этой возможностью. В плане "конкурентной подлости" было бы интересно наказать конкурентов в виде неожиданных 9999 посещений с разных прокси-серверов :-). Но я побоялся, что меня не поймут и начислят за нанесенный за один проход от $9999*0,05$=500 ущерб... В общем, не пожалеют.

А ТЕПЕРЬ НЕМНОГО ПОДЫТОЖИМ

Остается добавить, что все исходные файлы лежат на диске журнала или на сайте download.net.ru - заходите в гости.

Конечно, для нормальной продажи программы необходимо еще много всего. Например, придется вести техническую поддержку своего продукта. Отвечать на письма типа "Хочу приобрести Вашу программу для создания меню. Хотелось бы узнать...", собирать деньги, искать способы привлечения еще большего количества людей. Но это будет потом. А сейчас ты имеешь возможность самостоятельно и с достоинством выполнить директиву ВВП по тайному плану "Электронная Россия".

В первые дни после регистрации программу скачивали по 50 раз в день.

Денис Колисниченко (dhsilabs@mail.ru)

«ДОКУМЕНТАЛЬНЫЙ» ПЛЮС

СОЗДАНИЕ ДОКУМЕНТАЦИИ ДЛЯ ТВОЕЙ ПРОГРАММЫ

Хорошая коммерческая программа не может поставляться без хорошей документации. Правильно написанная документация – это безусловный плюс твоего продукта. Может, ты заметил, что в некоторых обзорах, помимо функциональности и дизайна программ, оценивается также и документация? В этой статье мы поговорим о том, как написать хорошую документацию для твоей программы.

С ЧЕГО НАЧАТЬ?

Конечно же, начинать стоит с выбора формата для будущей справочной системы. Тут все зависит от операционной системы, для которой предназначена программа. Сейчас имеет смысл использовать один из следующих форматов:

- формат HTML – документацию в формате HTML прочитают пользователи любой операционной системы;
- формат PDF (Portable Document Format) – формат PDF тоже поддерживается всеми операционными системами;
- стандартный файл справки Windows (.hlp) – поддерживается всеми версиями Windows, начиная с Windows 95;
- компилированный файл справки (.chm) – поддерживается ОС Windows, начиная с версии Windows 98;
- Формат man – поддерживается ОС UNIX/Linux.

Подробнее рассмотрим каждый из вариантов.

Сделанную в HTML документацию смогут прочитать пользователи любой операционной системы – Windows, Linux, MacOS. Выложить на сайт такую документацию, "одев" в дизайн сайта – без проблем. Впоследствии из формата HTML с помощью вспомогательных программ сможешь конвертировать документацию в форматы .chm и man. Именно в силу этих причин я считаю, что сам текст документации лучше сначала писать в формате HTML. При написании документации в формате HTML используй формат HTML 3.2, воздержись от лишних "наворотов", лучше без стилей и обязательно без JS и VBS, чтобы потом не было проблем при конвертировании.

Документация в формате PDF не приветствуется. Во-первых, для ее просмотра нужно будет установить Acrobat Reader, что увеличит размер дистрибутива. А по правилам хорошего тона ты обязан включить его в состав дистрибутива: у тебя же коммерческая программа – вдруг у пользователя не окажется Acrobat Reader, тогда раздастся звонок в службу поддержки. А что если таких пользователей будет 1000? Во-вторых, для создания такой документации нужен Acrobat, который, в отличие от Acrobat Reader, не бесплатен. Документацию в формате PDF нужно использовать когда ты разрабатываешь мультиплатформенный продукт. Именно для этих целей и был разработан формат PDF, документация в нем будет отображаться одинаково на разных платформах: любой шрифт, картинку в любом формате можно включить в PDF-файл, и ты можешь быть уверенным на все 100%, что у пользователя он будет выглядеть так же, как и у тебя. В-третьих: размер PDF-файла. Он не такой уж большой, но может случиться, что хорошая документация в PDF-файле займет больше места, чем сама программа. При нынешних дисковых объемах это ничем страшным не грозит, но при загрузке программы при соединении в 33,6 Кб разница в 2 Мб + Acrobat Reader довольно ощутима.

Формат HLP (стандартный файл справки Windows) тоже лучше не использовать. Windows 95 уже практически не используется, а справка в формате CHM выглядит свежее и привлекательнее, чем в формате HLP. Для просмотра CHM-файлов нужна не столько Windows 98, сколько Internet Explorer версии 4 или выше, который обычно не входит в состав Windows 95. Если учитывать поправку о Windows 95, в состав дистрибутива нужно добавить справку в формате HTML – много места она не займет. При создании ярлыков в программной группе в программе инсталляции нужно создать два ярлыка для справочной системы - один для CHM-формата, другой для HTML. А в обработчик события, отвечающий за вызов справочной системы, нужно поместить код, который проверял бы версию Windows и запускал бы нужную справочную систему.

При создании справки для Linux-программы используют два формата: man и HTML. Первый формат – это традиция, которую вредно нарушать, а HTML примечателен тем, что можно создать внешне привлекательную справку, да еще и с картинками - как говорится, лучше один раз увидеть, чем сто раз услышать. Создание документации для Linux-программы – это отдельный разговор, но мы еще к нему вернемся, а пока о том, как нужно правильно писать документацию.

НА ЧЬИ ПЛЕЧИ ВОЗЛОЖИТЬ ЭТУ ЗАДАЧУ?

Писать самому или платить опытному техническому писателю, написавшему с десяток разных руководств? Если ты не чувствуешь в себе писательского таланта, лучше путь напишет опытный человек. Но имей в виду, что хорошая документация стоит довольно дорого и за нее тебе, возможно, придется выложить намного больше, чем стоит один экземпляр твоей программы. Например, цены на полное руководство твоей программы могут варьироваться от $500 до 1200, а может, и того больше – все зависит от сложности продукта. Программа может стоить всего $50 – тебе нужно будет сразу заплатить в десять раз больше (и это минимум). Могу предложить небольшой трюк: если твоя программа стоит $50, подними стоимость до $55-ти, и первые десять дистрибутивов окупят вложения в документацию. За документацию ты платишь один раз.

Если тебя такой вариант не устраивает и у тебя нет $500, при этом никто не ждет, пока они у тебя появятся, можешь потратить немного времени и написать документацию самостоятельно. Потом пусть ее прочитает кто-то другой. Если уж совсем плохо получится, нужно один раз выложиться на нормальную документацию. Пока будем писать сами, я постараюсь описать основные моменты создания системы помощи.

ПРАВИЛЬНАЯ ДОКУМЕНТАЦИЯ

■ С форматом справки мы уже определились - это HTML, поскольку его можно использовать в сыром виде или преобразовать в любой другой формат. Для написания документации лучше использовать самый обыкновенный текстовый редактор, например, "Блокнот", чтобы не было соблазна влепить пару спецэффектов, которые или не будут поддерживаться браузером пользователя, или не скомпилируются при создании CHM-файла.

СТРУКТУРА

■ Прежде всего потрать немного времени и на листочке набросай структуру твоей справки так, как она отображалась бы слева в окне системы помощи, например:

```
Программа версии 1.1.
        Описание программы
        Что нового в версии 1.1
        +Быстрый старт
        +Расширенные функции
        +Администрирование
        +Обновление
        ...
        Служба поддержки
```

Если тебе трудно написать структуру системы справки с ходу, в качестве разделов системы справки используй пункты меню твоей программы и опиши последовательно все меню - пункт за пунктом:

```
Программа 1.1
        Описание программы
        Что нового в версии 1.1
        -Меню
                +Меню Файл
                +Меню Правка
                +Меню Вид
                +Меню Отчет
                +Меню Сервис
                +Окно
        ...
```

Второй вариант - это описать какой-нибудь пример работы с программой. Например, если твоя программа предназначена для автоматизации рабочего места менеджера и если ее основными функциями являются формирование счета-фактуры, накладной, договора - так и описывай, в таком же порядке:

```
Программа 1.1
...
        -Работа с программой
                Формирование счета-фактуры
                Формирование накладной
                Формирование налоговой накладной
                Договора
                Список счетов
                Список накладных
                Склад
                Отчет по складу
        -Дополнительные функции
                Фильтры
                SQL-запросы
        ...
```

Думаю, этой информации будет вполне достаточно, чтобы набросать удобную структуру. И еще два совета относительно структуры.

❶. Не поленись и создай предметный указатель: очень помогает при поиске какой-нибудь функции в файле справки.

❷. Если модуль администрирования твоей программы является отдельным приложением, создай для него отдельный файл справки.

ПОЛНОТА И ЧИТАБЕЛЬНОСТЬ ДОКУМЕНТАЦИИ

■ Следующее требование к документации - ясность, полнота, понятность для начинающих пользователей: нужно, чтобы они знали, что делает та или иная функция этой программы, но им совсем не обязательно знать технические подробности. В общем, смотри чтобы не получилось как в анекдоте:

- Ко мне не доходят сообщения.
- А вы прочтите их еще раз :-) .

Старайся сделать так, чтобы твоя документация была написана понятным языком. Пример неправильно написанного фрагмента документации: "Если не работает функция GetObjAddr, зарегистрируйте библиотеку comdlg32o.dll и пропишите ее в файле proga.ini.". Что здесь не так? Вроде все понятно. Будь это shareware или freeware-программы, такой текст бы прошел. Но коммерческая программа - совсем другое дело. Прочти этот пример еще раз и еще раз. А теперь представь лицо начинающего пользователя, который только научился дискету вставлять куда положено, когда он видит сообщение об ошибке, а потом пытается найти ответ в справке и видит этот текст. Представил? При большом количестве пользователей твой телефон покраснеет от входящих звонков. И ты, вместо того чтобы исправлять ошибки в своей программе или разрабатывать к ней полезные плагины, будешь консультировать и пользователей, и администраторов, разъясняя написанную документацию, а это, поверь, не очень благодарная работа.

Поэтому нужно писать ясно, чтобы не было двойственных ассоциаций и чтобы написанный текст не порождал больше вопросов, чем было до его прочтения. А в этом случае так оно и есть. Читаем: "Если не работает функция GetObjAddr". Откуда пользователь знает, работает она или нет. Лучше укажи точное название ошибки и диагностическое сообщение, которое увидит пользователь. А еще лучше промоделировать ситуацию и сделать скриншот этой ошибки, который потом можно "вставить" в документацию.

Читаем дальше: "зарегистрируйте библиотеку comdlg32o.dll". Это предложение порождает вполне обоснованный вопрос пользователя: "А как это сделать?" Будь уверен: в этом случае он к тебе точно позвонит, чтобы узнать, как зарегистрировать эту библиотеку. И еще: а что если пользователь не знает, что такое библиотека. Встречаются и такие экземпляры. Поэтому правильнее написать: "Выберите команду меню "Пуск" -> "Выполнить" и выполните команду":

```
regsvr32 comdlg32o.dll
```

Старайся употреблять поменьше технических терминов. А если употребляешь их, обязательно вставляй примечание, в котором объясняешь что к чему. Только избегай в определении других технических терминов, неизвестных пользователю, а если и от этого не сможешь удержаться, объясни их. Так повторяй до тех пор, пока не останется ни одного непонятного пользователю. Если примечаний наберется очень много - выдели их в отдельный файл, который назови глоссарием.

Вообще вариант с глоссарием самый лучший - ты сможешь свободно писать текст, не обращая внимания на термины, и объясняться понятным тебе и квалифицированному пользователю языком. Только помни о корректности: никакого сленга, только завершенные и понятные для всех формулировки. А потом прочти внимательно весь текст и все встречающиеся термины опиши в глоссарии. В этом случае и системные администраторы, и опытные пользователи не будут смотреть на твою документацию как на учебник по информатике, а начинающие пользователи всегда смогут найти неизвестный термин в глоссарии.

Итак, после небольших трансформаций текст "Если не работает функция GetObjAddr, зарегистрируйте библиотеку comdlg32o.dll и пропишите ее в файле proga.ini." будет выглядеть так:

```
Ошибка 110: Недоступна функция
GetObjAddr[:COMDLG320].
```

Попробуйте выполнить следующие действия:

❶. Определите место нахождения файла COMDLG320.DLL. Обычно данный файл должен находиться в каталоге:

C:\Program Files\My Company\My Proga\Libs\COMDLG320.DLL

Чтобы убедиться в этом, выполните команду основного меню Windows "Пуск" -> "Найти файлы и папки". В окне поиска в поле "Имя" введите COM32DLGO.DLL. Когда файл будет »

найден, запомните или запишите, в какой папке он расположен. Если файл не найден, попросите администратора переустановить программу.

❼. Нажмите на кнопку "Пуск" и выберите команду "Выполнить". Появится окно "Запуск программы". В поле "Открыть" введите команду:

```
regsvr32 <папка>\comdlg32.dll
```

где <папка> - это имя папки, в которой расположен файл COMDLG32O.DLL

❽. Запустите "Блокнот" и откройте файл C:\WINDOWS\MY_PROGA.INI. В конец этого файла добавьте строку:

```
COMDLL32O.DLL
```

Если ошибка не исчезла, обратитесь в службу поддержки.

Если ты заметил, вышеприведенный текст написан в стиле

```
Чтобы получить X, выполните
A)
B)
C)
```

Это неплохой стиль, и если ты будешь его придерживаться, к тебе будет обращаться меньше пользователей. А теперь вернемся к требованиям, а именно к полноте документации, которой нужно добиваться ради любви к тебе со стороны квалифицированных пользователей и администраторов. Первые из них хотят "копнуть поглубже", то есть полностью освоить и всесторонне разобраться в программе. В отличие от "продвинутых" пользователей, администраторами руководит не энтузиазм, а их прямые служебные обязанности. Опиши не только функции, но и ошибки, диагностические сообщения твоей программы. Очень подробно должны быть описаны настройки и опции программы. Избегай недокументированных функций: ты же не Microsoft? При описании модуля администрирования старайся чтобы он был описан наиболее полно: это самое важное. Если администратор не поймет, что так и к чему, то на пользователя вообще можно не надеяться.

ФОРМАТИРОВАНИЕ

■ Есть один прекрасный способ проверить форматирование твоей документации. Встань из-за компьютера и отойди от монитора на один метр. Ты должен видеть главную информацию. Если все слилось в одно пятно, попробуй сделать следующее:

❶. увеличить количество абзацев;

❼. увеличить интервал между абзацами;

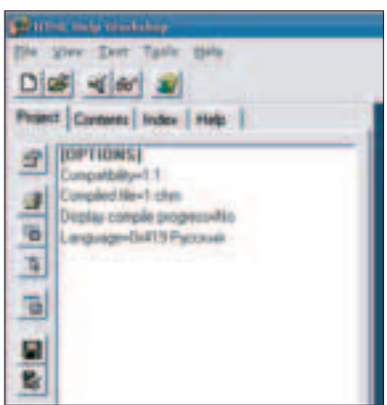
❽. выделить термины, команды, названия файлов и каталогов;

❶. увеличить размер шрифта.

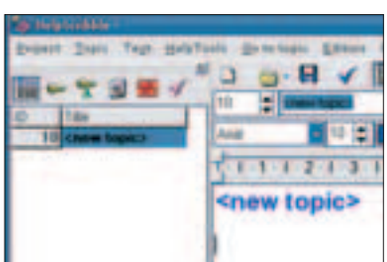
Для выделения терминов и команд я рекомендую использовать шрифт Arial - он более читабельный, чем Times.

ПРОГРАММЫ

■ Для создания HTML-документации вполне хватит "Блокнота", а какую программу будем использовать для создания CHM-файла? Могу порекомендовать две очень удобные программы для создания CHM-файлов: Microsoft HTML Help Workshop и HelpScribble.

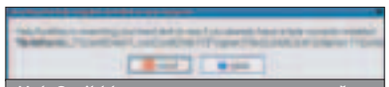


Microsoft HTML Help Workshop



HelpScribble

Первая очень удобна, позволяет из готовых HTML-файлов скомпилировать CHM-файл (Help Workshop позволяет). Лично мне эта программа симпатична больше (точнее, к ней я больше привык), чем Help Scribble, но это только мое мнение. Help Scribble любезно избавит тебя от необходимости использовать дополнительные программы (кроме компилятора справки) для создания файла справки, чем выделяется среди себе подобных. Текст документации можно создавать прямо в Help Scribble, причем в твоем распоряжении будет полноценный RTF-редактор, позволяющий изменять форматирование, применять макросы форматирования (аналог стилей в Word), вставлять картинки, кнопки и т.д. Главное отличие HTML Help Workshop от Help Scribble в том, что первая программа сама является компилятором CHM-файлов, а вторая - это RTF-редактор с функциями создания справки, но для самого процесса компиляции нужен внешний компилятор. Запусти Help Scribble и выбери команду меню



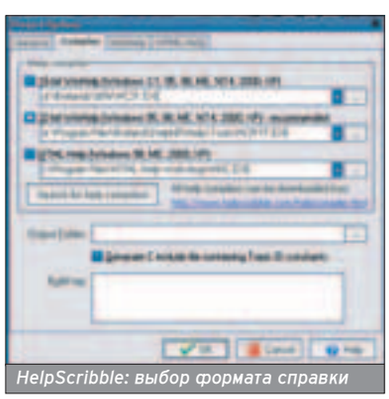
HelpScribble: поиск компилятора файлов справки

Project -> Options: ты увидишь окно поиска компилятора help-файлов.

Если ты хочешь создавать CHM-файлы с помощью Help Scribble, тебе все равно нужно установить HTML Help Workshop, поскольку он является компилятором CHM-файлов. Зато Help Scribble, в отличие от HTML Help Workshop, умеет создавать файлы справки любых форматов, а не только CHM. Например:

- 16-разрядный help-файл, который поддерживается всеми ОС Windows, начиная с Windows 3.11 и заканчивая Windows XP.
- 32-разрядный help-файл, который поддерживается операционными системами Windows 95-XP.
- CHM-файл, поддерживаемый системами Windows 98, ME, 2000, XP.

Возможности по работе с форматами в Help Workshop ограничиваются только конвертированием из старого (.HLP) формата в новый (.CHM).



HelpScribble: выбор формата справки

Где же взять компиляторы файлов справки? Можно стянуть с сайта Help Scribble: www.helpscribble.com/helpcompiler.html.

А можно поискать у себя дома на компактах – я уверен, что все они у тебя есть. Посмотри на окно опций Help Scribble: 16-разрядный компилятор нашелся в составе Borland C, 32-разрядный поставлялся вместе с Delphi, а компилятор для CHM-файлов – это HTML Help Workshop, который тоже можно найти на компакте с Delphi, хотя он не устанавливается по умолчанию. Если последнего у тебя нет, можно скачать его с сайта Microsoft (www.microsoft.com/workshop).

КОМПИЛИРОВАНИЕ CHM-ФАЙЛОВ

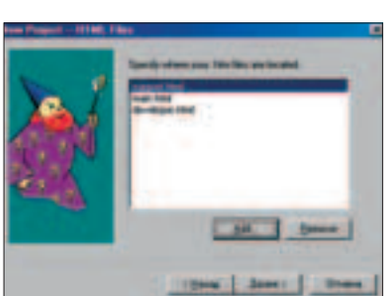
■ Теперь перейдем к практике. Перед компилированием CHM-файла нужно будет создать документацию в формате HTML. После того как набор HTML-файлов будет приготовлен, просмотри его в Internet Explorer. В других обозревателях просматривать файлы справки надобности нет, поскольку для просмотра компилированных CHM-файлов используется именно Internet

Explorer. Если нужно, исправь допущенные ошибки: для компилирования CHM-файлов нужно использовать уже чистовой вариант документации, поскольку исправить откомпилированный файл не получится.

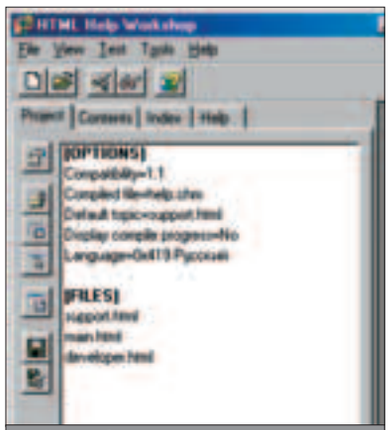
Не нужно размещать HTML-файлы, относящиеся к разным разделам, в отдельных подкаталогах. То же касается картинок, которые ты будешь использовать в своем проекте, - не нужно по привычке создавать для них каталог images. При создании web-сайтов плоская модель считается плохим тоном, однако это не относится к созданию справочной системы. Пользователю от этого ни лучше ни хуже не станет, а тебе дополнительные подкаталоги могут добавить проблем при компилировании CHM-файла. Запусти Help Workshop и выполни команду меню File, New. Из появившегося меню выбери Project. С помощью Workshop можно создавать не только проекты, но и HTML-файлы, текстовые файлы, файлы указателей, которые впоследствии могут быть добавлены в проект. Однако я предпочитаю готовить все вручную (в "Блокноте"), а потом только компилировать все вместе.

После выбора операции создания проекта будет запущен мастер, с помощью которого можно или конвертировать старый файл справки в формате HLP, или создать новый проект. Файл справки (CHM-файл) нужно сохранить в том же каталоге, где расположена документация в формате HTML. Если этого не сделать (сохранить файл в другом каталоге), начнется неразбериха с гиперссылками в процессе компилирования. Например, если справочная система расположена в каталоге C:\Projects\Zarp2\Help, то имя файла справки должно выглядеть так: C:\Projects\Zarp2\Help\help. Разумеется, вместо help ты можешь ввести любое другое имя файла (будет создан файл с расширением .HHP). На следующем шаге мастер спросит, имеются ли уже готовые файлы содержания (table of contents), указатель (Help index), файлы HTML. Нужно выбрать только последний пункт: документация в формате HTML уже готова. Откроется окно, в котором ты сможешь добавить нужные файлы.

Help Workshop создаст новый проект (файл с расширением .HHP). В окне Help Workshop ты увидишь закладки Project, Contents, Index и Help. Project предназначена для изменения общих параметров проекта, закладка Contents - для создания файла содержания, Index - для создания файла указателя. Закладку Help настоятельно рекомендую не использовать.

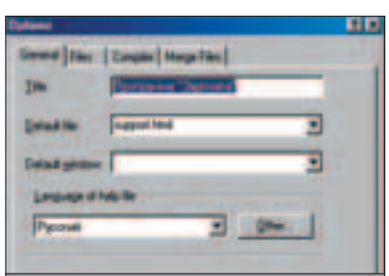


Выбор HTML-файлов

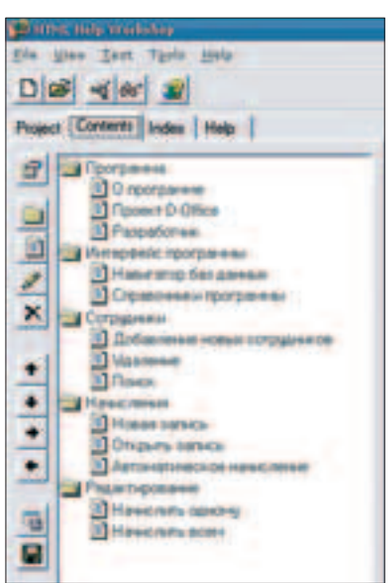


Файлы добавлены в проект

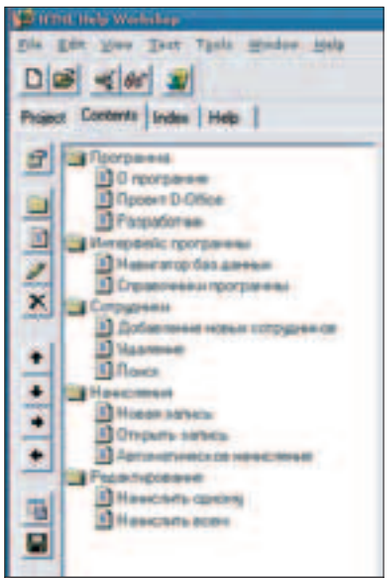
Обычно Help Workshop неправильно устанавливает начальную страницу (Default Topic). Эта страница будет отображена при запуске справочной системы первой. Нажми кнопку Change project options (самая верхняя кнопка на странице Project) и в появившемся окне введи заголовок проекта и выбери файл по умолчанию (Default file) из списка. Файл должен быть указан в списке FILES.



Начальная страница



Разделы справочной системы

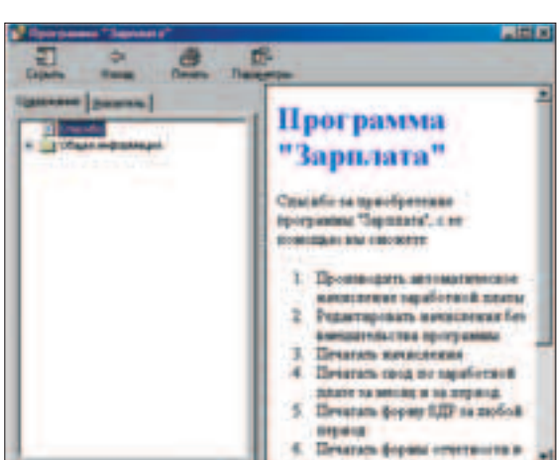


Отчет

Закладка Contents используется для создания нового файла содержания, который содержит информацию о разделах справочной системы. Закладка Index позволяет создать файл указателя – не ленись! Разобраться с этими закладками особого труда для тебя не составит, поэтому комментировать их не буду.

Когда файл проекта будет готов к компиляции, нажми кнопку Compile HTML file, которая находится на панели инструментов Help WorkShop, или выбери команду File, Compile. Убедительно рекомендую перед компилированием сохранить файл проекта. После завершения процесса компилирования увидишь протокол компилирования, в котором Help WorkShop предоставит подробный отчет о разделах, файлах, картинках, ошибках и предупреждениях.

Вот, собственно, и все. Помни: конечный пользователь не хочет много думать, он хочет получить простой, понятный и наглядный продукт. Документация - неотъемлемая часть хорошей программы.



Готовый CHM-файл

Денис Колисниченко (dhsilabs@mail.ru)

СПЕЦ ПО УСТАНОВКЕ

СОЗДАНИЕ ИНСТАЛЛЯТОРА ДЛЯ ТВОЕЙ ПРОГРАММЫ

Ни один серьезный продукт не обойдется без инсталлятора – специальной программы, которая распакует дистрибутив, скопирует файлы твоей программы в выбранный пользователем каталог, создаст программные группы, внесет необходимые изменения в реестр, а также установит и зарегистрирует необходимые твоей программе библиотеки.

ВЫБИРАЕМ СОФТ

Ясно, что своими руками писать инсталлятор мы не будем, поскольку это лишние затраты времени, которые никто не оценит. Разве что у тебя настолько уникальная программа, что ни один инсталлятор не сможет ее правильно установить, поскольку "не знает" всех ее особенностей. Тогда эта статья не для тебя, поскольку в ней будут рассмотрены популярные стандартные решения для создания инсталляторов, а именно InstallShiled для Delphi, Visual Studio Installer, Setup Factory и InnoSetup. Конечно, твой выбор не ограничивается этими программами, но они были выбраны как одни из самых лучших в своем классе.

Скорее всего, ты когда-нибудь использовал одну или несколько программ для создания инсталляторов. Если ты использовал две-три или более таких программ, то, наверное, замечал, что эти программы бывают трех типов: "автоматизированные", "ручные" и "полуавтоматизированные". Первые представляют собой мастер (или его пошаговое подобие с возможностью выбирать шаг создания инсталлятора), создающий проект инсталлятора. Примером программ такого типа может стать InstallShield для Delphi. Второй тип программ подразумевает создание разработчиком (то есть тобой) специального скрипта, на основании которого будет создан инсталлятор. Скрипт описывает этапы установки программы и их последовательность. Примером второго типа программ может послужить не рассмотренная в статье программа WISE. Третий тип программ - это своеобразный симбиоз первого и второго типов. Программы третьего типа обладают собственным мастером, который "пишет" скрипт. Ты можешь откомпилировать этот скрипт сразу или немного изменить его, подправив, например, ключ реестра или другую информацию, а потом уже компилировать. Большинство современных программ для создания инсталляторов относятся именно к третьему типу.

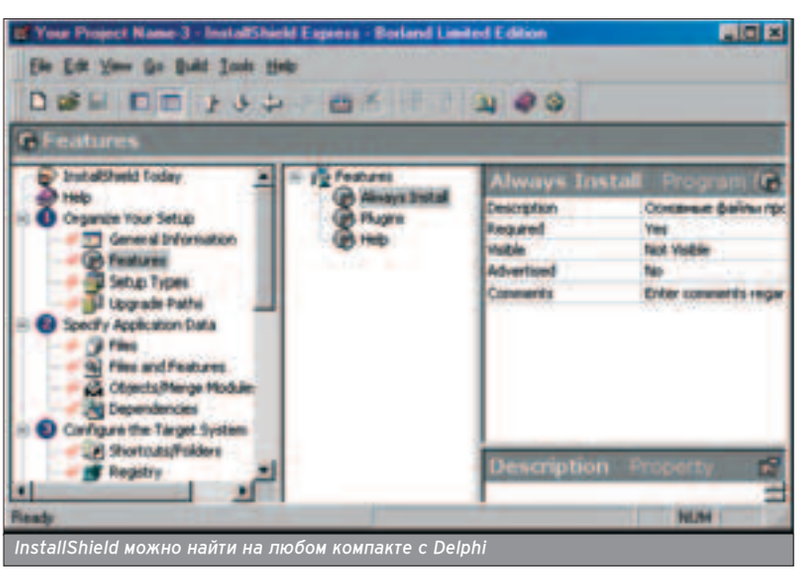
Наиболее удобны программы первого типа. С их помощью можно создать инсталлятор, как говорится, "за пару кликов мышью". Но такие программы могут не предусмотреть всех твоих пожеланий, и вполне возможно, что при их использовании тебе будет не хватать некоторых функций. Программы второго типа более функциональны, чем программы первого типа - тут есть где разгуляться. Их недостаток - то, что фактически ты сам пишешь программу-инсталлятор. Время экономится только на создании стандартных диалогов выбора каталога для установки, окна чтения лицензии, окна выбора компонента программы и т.д. Эти окна создает сама программа. Но что будет делать твой инсталлятор - это уже как ты сам напишешь. На создание инсталляторов с помощью программ второго типа уходит намного больше времени, чем в предыдущем случае.

Третий тип программ сочетает в себе все преимущества первых двух типов. В твоем распоряжении мастер, создающий сценарий установки. Если твоя программа стандартная, то есть не требует выполнения каких-либо специфических операций при установке, инсталлятор будет создан через пару минут, как будто ты используешь полностью автоматизированную программу первого типа. Но если тебя что-то не устраивает, можешь отредактировать созданный программой сценарий, как будто ты используешь программу второго типа.

INSTALLSHIELD

А теперь перейдем к рассмотрению названных программ. Первая на очереди - InstallShield, которая абсолютно бесплатна, ее можно найти на любом компакте с Delphi, желательно, чтобы версия Delphi была посвежее - на новом компакте версия Install Shield тоже будет относительно свежая. Если компакта под рукой не окажется, ты всегда сможешь стянуть ее с сайта Borland. Вот что входит в "обязанности" этой программы:

- создание стандартных диалогов инсталлятора;
- создание нескольких типов установки (например, Standard, Compact и Custom);
- удобная работа с реестром (ты можешь указать, какие изменения в реестре нужно сделать при установке программы);



InstallShield можно найти на любом компакте с Delphi

- удобная работа с библиотеками;
- поддержка BDE (если твоя программа работает с BDE, можно ни о чем волноваться, а просто выбрать BDE, и InstallShield включит его в состав твоего дистрибутива, также будет нужным образом настроен BDE);
- создание ярлыков, папок в меню "Пуск";
- поддержка носителей разного размера – от флоппи до DVD.

Это, конечно, не все функции программы, но основные возможности, как говорится, "учтены". Использовать InstallShield очень удобно, если твоя программа использует BDE. Во всех остальных программах тебе придется описывать BDE "вручную", что, мягко говоря, неудобно. Даже если в InstallShield нет очень нужной функции, это с лихвой компенсируется поддержкой BDE. Ты даже не представляешь себе, от какой головной боли избавишься, если будешь использовать InstallShield для установки и конфигурирования BDE.

VISUAL STUDIO INSTALLER

Visual Studio Installer – это бесплатная утилита от Microsoft, предназначенная для создания инсталляторов программ, написанных с помощью Visual Studio. Программа доступна в двух версиях, 1.0 (для Win 98) и 1.1 (для Win ME, 2000, XP), по адресу: http://msdn.microsoft.com/vstudio/downloads/tools/vsi11/default.asp.

Перед установкой программы убедись, что у тебя есть Visual Studio Service Pack 3. Главное преимущество Visual Studio Installer связано именно с самим Visual Studio. Если твоя программа написана на Visual Basic или Visual C, Visual Studio Installer сам "пропишет" в проекте инсталлятора все нужные библиотеки, которые потом будут "скомпилированы" в дистрибутив. Кроме этого, Visual Studio Installer ни в чем особенном не проявляет себя – программа как программа.

SETUP FACTORY (WWW.SETUPFACTORY.COM)

Setup Factory – это коммерческий продукт, предназначенный для создания инсталляторов. Стоит немалых денег – за седьмую версию просят $395. Ясно, что никто из нас эти деньги платить не собирается, но все равно сумма говорит за себя. Дистрибутив Setup Factory 7.0 весит 18 Мб и постоянно доступен для загрузки :). Программа может работать под управлением Windows 95/98/Me/NT/2000/XP/2003 Server. Конек Setup Factory, который будет непременно оценен пользователем, - это темы диалогов. Программа содержит по умолчанию более 20-ти различных тем стандартных диалогов инсталлятора, поэтому твой продукт не будет похож на решения конкурентов.

Setup Factory – это программа третьего типа, то есть мастер плюс возможность редактирования сценария, хотя больше ориентирована именно на создание сценария вручную. В этом случае тебе открываются более 250-ти разных функций программы, которые недоступны при использовании мастера.

Основные функции:

- создание единственного файла дистрибутива - большого и огромного setup.exe (конечно, если тебе нужно разбить дистрибутив на дискеты или компакт-диски, программа сделает это);
- поддержка серийных номеров, в основе которых лежит всем известный алгоритм MD5;
- поддержка dat окончания действия дистрибутива (очень полезно для shareware-продуктов);
- поддержка редактирования реестра;
- поддержка различных языков;
- создание ярлыков и программных групп;
- поддержка проектов Visual Basic;
- вывод отчетов о проекте в формате HTML;
- полный uninstaller;
- проверка правописания;
- автоматический ("тихий") режим установки, в котором пользователю не задается вопросов и программа устанавливается сразу после запуска setup.exe.

INNO SETUP (WWW.INNOSETUP.COM)

Inno Setup – бесплатная программа для создания инсталляторов, причем ее исходный код (написан на Delphi) доступен в Сети.

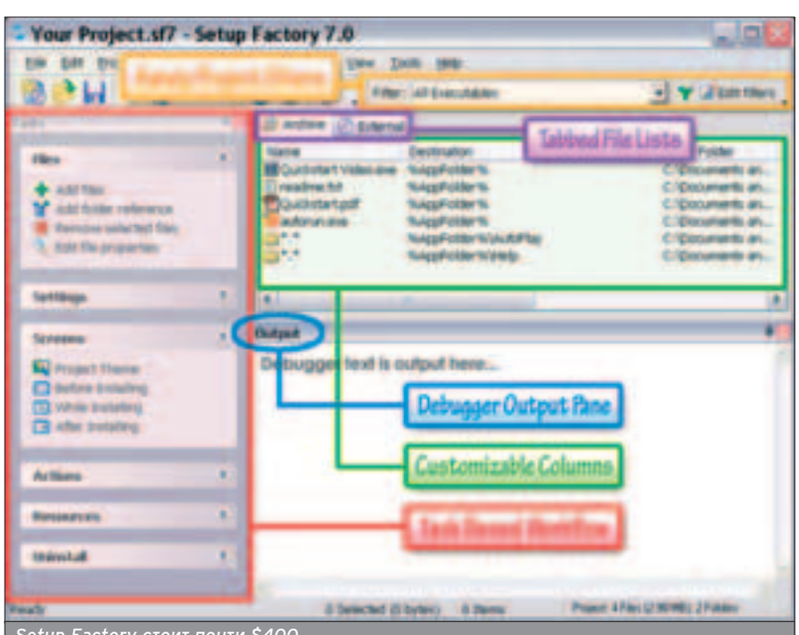
Основные функции программы:

- поддержка всех 32-битных версий Windows – Win 95, 98, ME, NT 4.0 SP6, 2000, 2003, XP;
- способность программы "упаковать" весь дистрибутив в единственный файл setup.exe или разбить дистрибутив на диски;
- различные типы установки: полная, минимальная, выборочная;
- полный uninstaller;
- сжатие дистрибутива в основном методом сжатия 7-Zip LZMA;
- умение инсталлятора сравнивать версии файлов, заменять уже установленные файлы с более старой версией, регистрировать DLL/OCX и библиотеки типов, а также устанавливать шрифты;
- создание ярлыков и программных групп;
- работа с реестром и INI-файлами;
- поддержка редактирования созданного мастером сценария;
- поддержка многоязыковых инсталляций;
- "тихая" установка и удаление программы.

Программа обладает всеми необходимыми функциями, Inno Setup - очень простой и удобный инсталлятор. Кроме того, этот универсальный инсталлятор не привязан ни к Delphi, ни к Visual Studio. Плохо это или хорошо, зависит от твоей программы. Лучше, конечно, чтобы InnoSetup поддерживал BDE и библиотеки Visual Studio, но пока этого нет.

КАКУЮ ПРОГРАММУ ВЫБРАТЬ?

Ты не знаешь, какую программу выбрать? Поначалу используй стандартный инсталлятор, который "идет в нагрузку" с твоей IDE. Если ты пишешь на Delphi, попробуй сначала использовать InstallShield, а потом уже пробовать другие инсталляторы. Если же ты пишешь на Visual C/Visual Basic, используй Visual Studio Installer. Если тебе будет не хватать возможностей стандартных программ, попробуй сначала InnoSetup, а потом SetupFactory. Единственный случай, когда у тебя не будет выбора, это если твоя программа использует BDE: тогда намно- »



Setup Factory стоит почти $400

го рациональнее использовать InstallShield.

INNOSETUP: СОЗДАНИЕ ДИСТРИБУТИВА

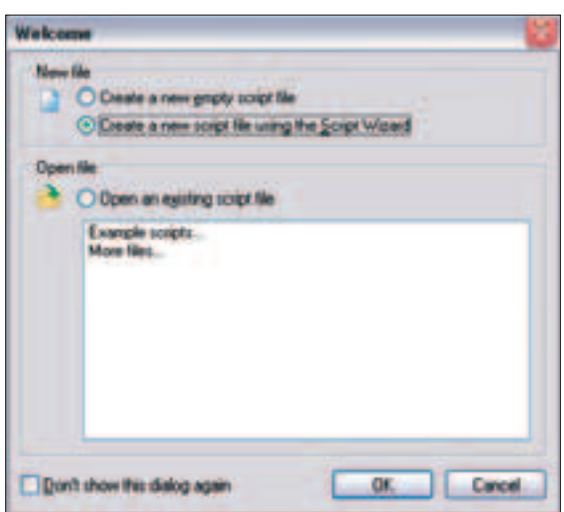
Попробуем создать дистрибутив с помощью InnoSetup. Программа не использует BDE, поэтому весь дистрибутив будет состоять всего из четырех файлов: sto.exe, sto.chm, sto.ini, vincheck.exe. Поехали! Запускаем InnoSetup и выбираем Create a new script file using the Script Wizard.

Появится окно мастера, в котором нужно будет ввести информацию о программе.

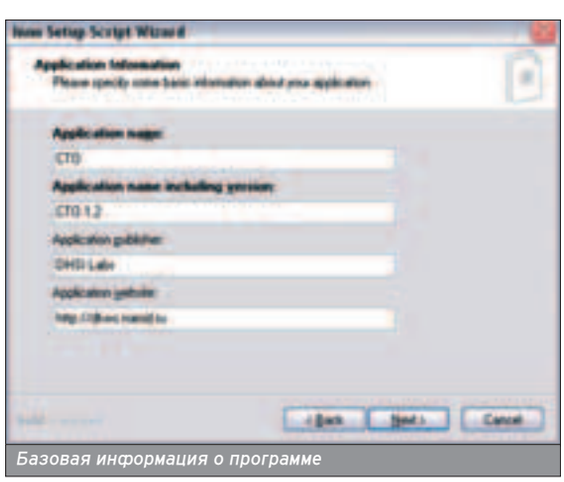
Следующий шаг мастера – выбор каталога для установки программы. Выбор небольшой:

- Program Files directory – программа будет установлена в каталог Program Files на одном из логических дисков;
- (Custom) – по твоему усмотрению.

Каталог для твоей программы, который будет создан в каталоге Program Files, задается в поле Application directory name. Обычно он устанавливается по имени программы. По умолчанию включена опция Allow user to change the application directory, позволяющая пользователю изменять каталог установки. За исключением редких случаев выключать ее не нужно.



Создание нового сценария



Базовая информация о программе

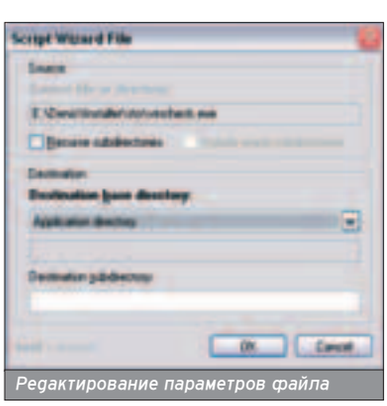
Теперь выбираем файлы, которые нужно включить в состав дистрибутива. В поле Application main executable file указывается основной исполнимый файл приложения. Потом с помощью кнопок Add files(s)/Add directory добавляются нужные файлы/каталоги.

Кнопка Edit позволяет подправить параметры файла, а именно, изменить каталог назначения файла – больше ничего полезного она не делает. В качестве каталога назначения доступны следующие каталоги:

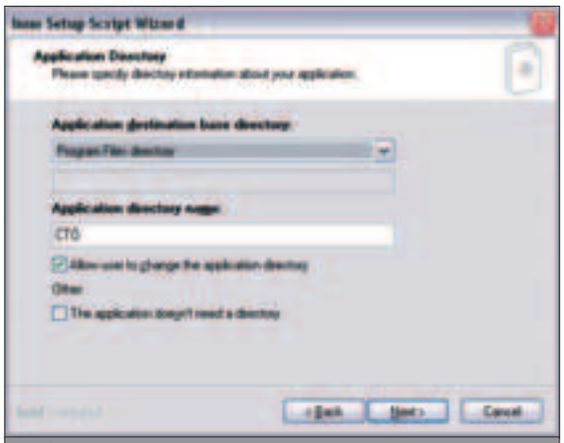
- Application directory – каталог приложения, то есть каталог, в которой устанавливается программа;
- Program Files directory – каталог Program Files;
- Common Files directory – каталог Common Files;
- Windows directory – каталог %WINDIR%;
- Windows system directory – каталог %WINDIR%\System;
- Setup source directory – каталог, в котором находился файл инсталлятора setup.exe;
- System drive root directory – корневой каталог системного диска, например, C:\;
- Common Startup folder – общая папка автозагрузки (программа будет запускаться автоматически для всех пользователей);
- User Startup folder – пользовательская папка автозагрузки;
- Custom – на твое усмотрение.

В поле Destination subdirectory можно указать подкаталог каталога назначения.

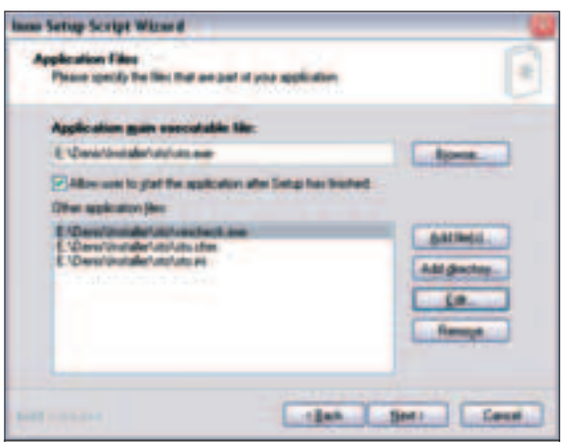
Следующий шаг позволяет указать имя создаваемой программной группы, а также то, какие ярлыки нужно создавать.



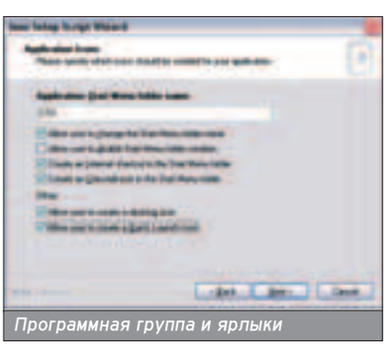
Редактирование параметров файла



Выбор каталога для установки



Добавление файлов

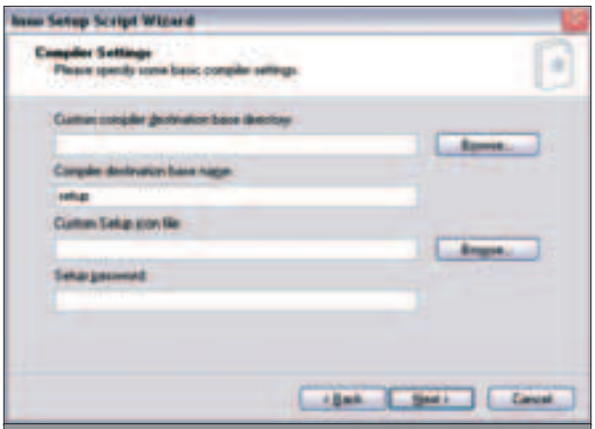


Программная группа и ярлыки

Жаль, что мастер не позволяет создавать ярлыки для отдельных файлов, поэтому, чтобы в программной группе было больше ярлыков, выбери создание ярлыка для твоего сайта (Internet shortcut), а также ярлыка для uninstaller'а. Можно, конечно, подправить сценарий установки, но лучше если бы мастер программной группы был более функциональным, например, как в InstallShield.

После создания программной группы мастер предложит указать файл лицензии и файлы, которые будут отображены до и после инсталляции. Следующий шаг очень важен: ты можешь указать другой компилятор, выбрать имя для инсталлятора (обычно setup), пиктограмму для setup.exe, а также установить пароль для запуска setup.exe.

Все, что тебе осталось, - это нажать кнопку Finish. Затем программа спро-



Параметры компилятора

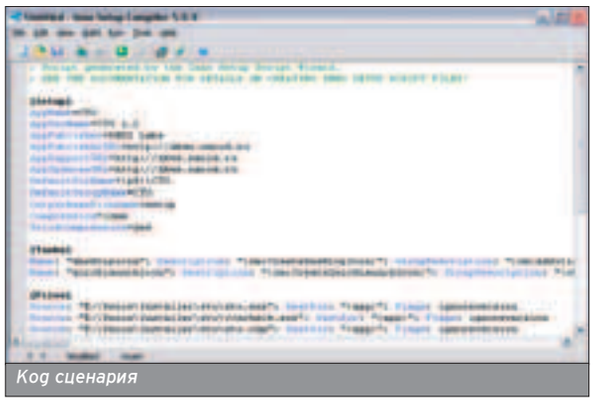
сит, нужно ли откомпилировать сценарий прямо сейчас: не нужно, лучше просмотри код сценария.

Разобраться с только что созданным сценарием несложно, тем более что, прочитав справку, ты сможешь не только понять, что там написано, но и усовершенствовать сценарий. Теперь нажимаем кнопку Run или жмем F9 - наш сценарий будет откомпилирован и запущен.

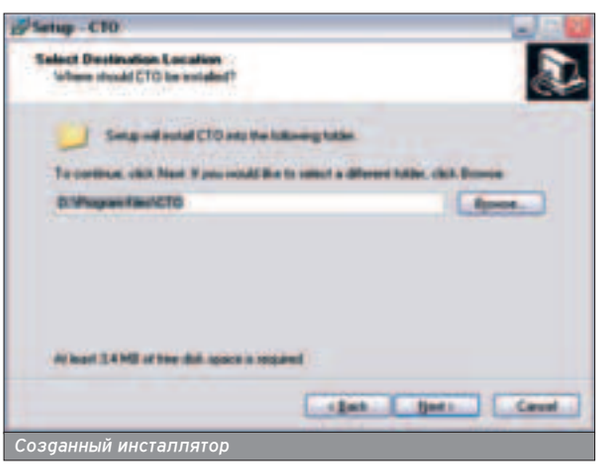
Смотрим протокол, чтобы найти только что созданный setup.exe. Протокол отображается в нижней части окна программы".

```
[10:09:15] --- Setup started
[10:09:15] Setup version: Inno Setup version 5.0.8
[10:09:15] Original Setup EXE: E:\Program Files\Inno Setup 5\Output\setup.exe
[10:09:15] Setup command line: /SL4 $10038E "E:\Program Files\Inno Setup
5\Output\setup.exe" 679818 51712 /DEBUGWND=$102E2
[10:09:15] Windows version: 5.01.2600 SP1 (NT platform: Yes)
[10:09:15] User privileges: Administrative
...
```

Суммарный размер четырех файлов дистрибутива - 2,7 Мб, размер файла setup.exe - чуть больше 900 Кб. Вполне приемлемо, не так ли? Дерзай.



Код сценария



Созданный инсталлятор

Content:

Крис Касперски ака мыщъх

НА ЧЕМ И КАК

РАЗБИРАЕМСЯ: КАК ПРАВИЛЬНО ПИСАТЬ БОЛЬШУЮ И КАЧЕСТВЕННУЮ ПРОГРАММУ

Современные программные комплексы уже не программируются на одном языке, а представляют собой конгломерат гибридного типа. Правильный выбор технологии программирования очень важен. Могу сказать без преувеличения, что он определяет судьбу продукта, сроки реализации, трудоемкость разработки, расширяемость и т.д. Оставь священные войны "Pascal vs C" или "C vs Ассемблер". Перед нами стоит вполне конкретная задача - создать конкурентоспособный продукт, который должен иметь успех на рынке.

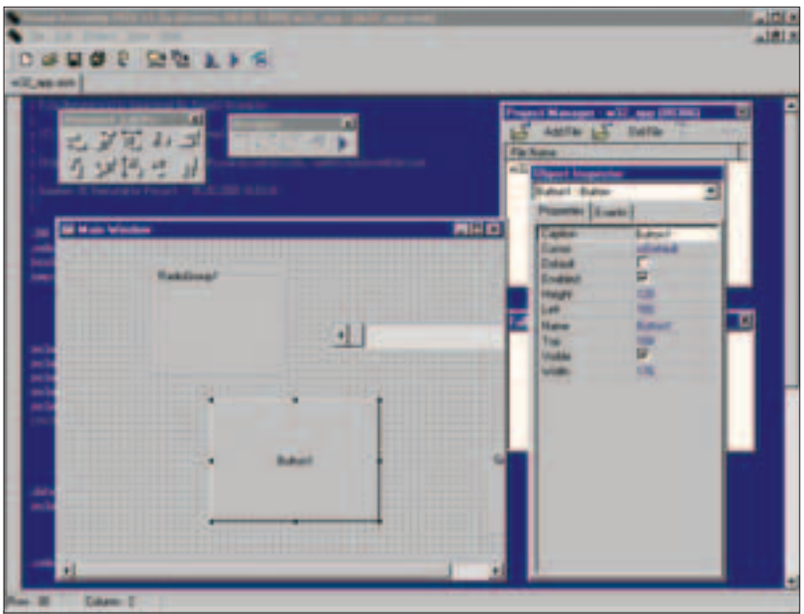
НА ЧЕМ ПИСАТЬ?

Для многих основной язык разработки - это C, в "конституции" которого декларирована приближенность к аппаратуре (а значит, и высокая эффективность). Фактически это МЕГАассемблер, предоставляющий программисту полную свободу. Изначально ориентированный на чисто "хакерские" цели, C завоевал признание миллионов программистов. Где он только не используется: низкоуровневое и высокоуровневое системное программирование, встраиваемые системы, финансовые и научные расчеты, общее прикладное программирование etc. При всех присущих ему недостатках (сходи на SU.C-CPP, почитай Харона), это лучшее средство для выражения программистской мысли в наиболее естественной для него форме. Конструкции в стиле: x == (flag?sin:cos)(y) здесь вполне законны и являются нормой. В этом смысле C очень похож на спектрумовский Бейсик, везде, где это только возможно, допускающий подстановку выражений. Отсутствие встроенных средств для работы с массивами вкупе с доминирующей небрежностью проектирования приводит к многочисленным ошибкам переполнения (buffers overflow), а "демократичность" работы с указателями - к утечкам памяти. Писать сетевые приложения на C категорически не рекомендуется. Но все равно пишут. Отсюда берутся черви, атаки на удаленные системы и прочие коварства виртуального мира?

Сейчас структурное программирование считается достоянием истории. В моду вошел ООП. C++ завладел умами программистов. Объектно-ориентированный подход пропагандируется как единственно возможный способ программирования вообще, а на приверженцев классического C смотрят как на чудаков или недоучек. Прямо насилие какое-то получается! На самом деле преимущество ООП над процедурным программированием весьма спорно, и возложенные на него ожидания так и не оправдались. Ошибок не стало меньше, сроки разработки только возросли, удачных примеров повторного использования кода что-то не наблюдается, а требования к квалификации разработчиков взлетели до небес.

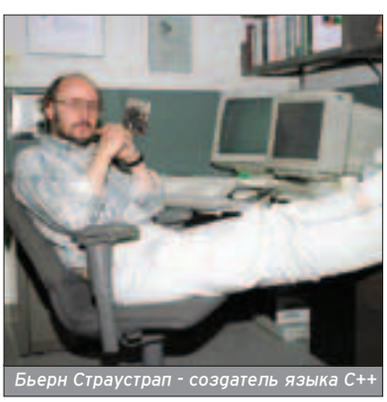
Но C++ поддерживает не одну, а целых три парадигмы программирования: структурное программирование в духе улучшенного C, абстрактные типы данных, позаимствованные из Ada ;), и, наконец, объектно-ориентированный язык в стиле Simula. Вот что говорит по этому поводу Бьерн Страуструп: "При создании программы всегда есть некоторое количество вариантов, но в большинстве языков выбор сделал за вас проектировщик языка. В случае C++ это не так: выбор за вами. Такая гибкость, естественно, не годится для тех, кто считает, что существует лишь один правильный способ действовать. Она может также отпугнуть начинающих пользователей и преподавателей, полагающих, что язык хорош, если его можно выучить за неделю. C++ к таким языкам не относится. Он был спроектирован как набор инструментов для профессионалов, и жаловаться на то, что в нем слишком много возможностей, значит уподобляться дилетанту, который, заглянув в чемоданчик обойщика, восклицает, что столько разных молоточков никому не понадобиться".

Приплюснутый C - это целый мир. Богатый ассортимент языковых возможностей еще не обязывает пользоваться ими. Объектный подход бесполезен в драйверах. Сколько программисты ни пытались найти ему применение - так и не получилось, а вот парадигму "улучшенного C" (объявление переменных по месту использования, а не в начале программы и т.д.) используют многие. Правда, в драйверах (равно как и в модулях сопряжения со средой) жесткая типизация приплюснутого C порождает дикий кастинг (явное преобразование типов), уродуя код и отнимая массу времени. Автоматической сборки мусора в C++ нет, а значит, от утечек памяти он не спасает (даже если используются "умные" указатели и прочие извращения). Механизмы для работы со строками переменной длины как будто бы появились, но переполнения буферов встречаются и до сих пор с завидной регулярностью. Так что C++ - это не панацея от всех бед, а всего лишь объект раздутой рекламной кампании. Страуструп оценивал количество пользователей приплюснутого C в 5 000 программистов по всему миру. Вряд ли он ошибался. Феноменальная популярность плюсов вызвана скорее высокой себестоимостью его компиляторов и вытекающей отсюда раскруткой (надо же как-то возвращать вложенное), чем техническими достоинствами.



Visual Assembly Pro - визуальный ассемблер под Windows с кучей мастеров и с удобным редактором диалогов (но TASMED от Solar Designer'a все равно круче)

Чистых компиляторов языка C уже давно не существует, сейчас они все поставляются вместе с плюсами. На одной лишь x86 платформе их насчитывается более десятка. Среди них есть как бесплатные, так и коммерческие, причем бесплатных значительно больше. По качеству кодогенерации лидируют Microsoft Visual C++ (входящий в состав халявных Platform SDK и Visual C Toolkit) и Intel C++ (версия под Windows условно-бесплатная, а под Linux - бесплатная для некоммерческого использования, однако никто не запрещает нам компилировать системно-независимые куски кода в объектные файлы под Linux и линковать их с Windows-приложениями). WATCOM C++, когда-то оптимизировавший круче всех, прекратил свое существование и теперь развивается в рамках проекта Open WATCOM, который, по свидетельствам очевидцев, больше глючит, чем работает. Borland C++ тоже бесплатен, однако с качеством кодогенерации у него кранты. Это худший оптимизирующий компилятор из всех! В мире *nix большой популярностью пользуется GCC, портированный в том числе и под Windows. Однако под окнами он чувствует себя неуютно, и особого резона в нем нет.



Бьерн Страуструп - создатель языка C++

Pascal, получивший второе рождение в среде Delphi, изначально задумывался как "студенческий" язык, который бы демонстрировал основные концепции структурного программирования. ООП в него перетащили уже потом, да и то криво. Получилось что-то вроде морской свинки. И не свинки, и не морской, зато от одного названия сдохнуть можно. Подход, исповедуемый Pascal'ем, находится где-то между Бейсиком и C, поэтому многие называют его "игрушечным" языком программирования. Но именно такой язык и нужен разработчикам интерфейсов! Не зря Borland остановила на нем свой выбор. Delphi намного удобнее появившегося вслед за ним C++ Builder (хотя тут можно и поспорить), но как бы там ни было, это коммерческий продукт, который хочет денежку. Приложения, разработанные в Delphi, с некоторыми ограничениями можно откомпилировать бесплатным транслятором Free Pascal, хотя для разработки с нуля Free Pascal непригоден, так как у него нет соответствующей IDE. То есть пригоден, конечно, но только не при визуальном подходе.

Остальные языки программирования используются намного реже. Java в основном применяется во встраиваемых системах, например, в тех же сотовых телефонах. В web-программировании до недавнего времени активно использовался Perl, но сейчас он начинает сдавать позиции, уступая PHP. Оба интерпретатора бесплатны, но их дальнейшая судьба под угрозой. На рынок сетевого программирования легла грозная тень надвигающейся эпохи .NET, за которой стоит Билл-разрушитель, а "Bill always wins". Базы данных, в России традиционно писавшиеся на Шкипере и Фоксе, сейчас реализуются на встроенных макроязыках типа Visual Basic'a, обслуживающего монстров вроде Access или Excel. Про 1C-бухгалетерию я уже и не говорю. Но ведь это тоже программирование! Пускай и в наиболее извращенной форме. В общем, языков много хороших и разных, и каждый из них может тебе в чем-то пригодиться ("Но не программируйте на Visual Basic, если вы можете этого избежать" (c) Дао программирования - прим. Горл).

КАК ПИСАТЬ

В правильно спроектированной программе можно выделить три независимых уровня: слой сопряжения со средой (оборудованием/операционной системой), "вычислительная" часть и пользовательский интерфейс. Эти уровни предъявляют различные требования как к языкам программирования, так и непосредственно к самим программистам. Обычно они реализуются разными людьми, образующими программистскую команду. Конечно, утилиту в несколько тысяч строк исходного кода можно сварганить и самостоятельно, но мы же не об этом сейчас говорим. Рассмотрим внутреннюю структуру типичного приложения (десятки и сотни тысяч строк исходного кода) во всех подробностях.

НА ГЛУБИНЕ

Слой сопряжения со средой абстрагирует программу от особенностей конкретного окружения, сокращая трудоемкость переноса на другие операционные системы и языки программирования. Если же заранее известно, что этого не потребуется, слой сопряжения со средой частично или полностью "вживляется" в вычислительную часть, что упрощает ее проектирование и программирование.

Для абстрагирования от операционной системы на все API-функции надеваются "обертки". Хорошим примером тому служит стандартная библиотека C - fopen и malloc работают и в Windows 3.x/9x/NT, и в *nix, и даже в MS-DOS, в то время как CreateFile и HeapAlloc - только в Windows 9x/NT. Тем самым C частично абстрагирует нас от операционной системы, а Delphi/C++ Builder идут еще дальше. Слагающие их библиотек образуют что-то вроде операционной системы в миниатюре, и изучать win32 становится необязательно (только чур я тебе этого не говорил!). Простой перекомпиляции достаточно, чтобы перенести программу на Linux, а с некоторыми ограничениями и на другие операционные системы.

По понятным причинам штатные библиотеки ориентированы на сравнительно небольшой круг стандартных задач, а все, что находится за его пределами, требует тесного взаимодействия с операционной системой. Прочитать сектор с диска, намылить приятелю шею, выдернуть лоток CD-ROM и т.д. Проблема в том, что в каждой версии Windows (не говоря уже о »

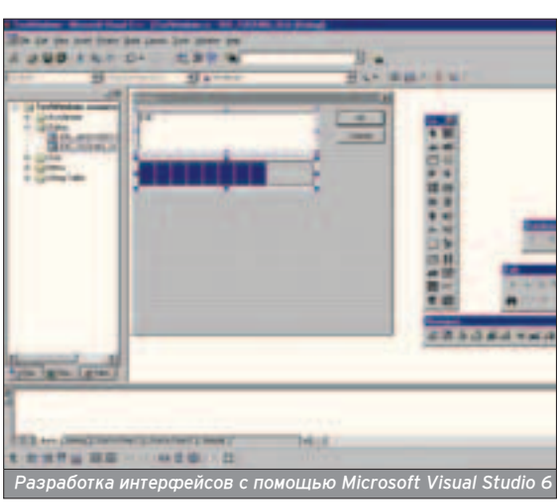
"В России учат быть гениями, но не учат быть просто профессиональными работниками". Давид Ян, компания ABBYY.

Диалог с формума:
- ...у C++ по сравнению с C намного больше плюсов!
- Ну да, ровно два.

FAR - самое правильное IDE с точки зрения системщика и продвинутого прикладника

*nix и полуоси) эти задачи решаются по-разному, часто с применением ассемблера и прочих хитрых хаков. Их разработка требует хорошего знания операционной системы, помноженного на высокую квалификацию, однако заниматься разработкой самому необязательно. Существует масса готовых библиотек (в том числе бесплатных), правда, качество реализации большинства из них оставляет желать лучшего, к тому же они тянут за собой неизвестное заранее количество подозрительных драйверов и вспомогательных библиотек, ожесточенно конфликтующих между собой. Но это уже издержки цивилизации. Либо создавай свой собственный код, либо используй то, что удалось найти в Сети.

Работу можно считать законченной, когда в вычислительной части не останется ни одного прямого вызова API-функции. Обертки в основном пишутся на C с редкими вкраплениями ассемблера и оформляются как DLL или статически компонуемый модуль. Опытные программисты используют одну и ту же библиотечку для всех своих проектов, поэтому DLL в большинстве случаев все же предпочтительнее (но не забывай, что большое количество явно прилинкованных DLL ощутимо замедляет загрузку приложения). Использовать C с плюсами



Разработка интерфейсов с помощью Microsoft Visual Studio 6

здесь в общем нежелательно. Во-первых, исходный текст получается избыточнее, а во-вторых, "манглеж" приплюснутых имен не стандартизирован. Динамическую библиотеку, скопированную Borland C++, к проектам Visual C++ подключить совсем не просто (впрочем как и наоборот). Pascal же здесь категорически непригоден: отсутствует нормальный интерфейс с операционной системой, это вынуждает ходить окружным путем, то есть через пятую точку.

Иногда возникает необходимость обратиться к защищенным ресурсам, доступным только из режима ядра (например, вызвать привилегированную команду процессора). Программисты старшего поколения в этом случае пишут псевдодрайвер. С точки зрения операционной системы псевдодрайвер выглядит как обыкновенный драйвер, но, в отличие от него, не управляет никакими устройствами, а просто выполняет привилегированный код, общаясь с прикладным приложением через интерфейс IOCTL. Псевдодрайверы пишутся на C с небольшой примесью ассемблера. Чистый ассемблер более элегантен, но экономически невыгоден. Приплюснутый C скрывает подводные камни. Не используй его, если точно не уверен, что именно ты делаешь. Для облегчения разработки драйверов фирма NuMega выпустила пакет Driver Studio, и хотя многие от него без ума, общее впечатление отрицательное. Лучше купи "Недокументированные возможности Windows 2000" Свена Шрайбера. Там ты найдешь готовый скелет псевдодрайвера, который легко адаптировать под свои нужды, не отвлекаясь на изучение посторонних дисциплин. Еще можно использовать готовые библиотеки, дающие с прикладного уровня доступ к портами и прочим системным компонентам (исходный текст одной такой библиотеки приведен в моей книге "Техника защиты лазерных дисков от копирования"). Естественно, такой подход небезопасен и совсем не элегантен.

Возникает дилемма: либо пыхтеть над DDK (тысячи страниц и все на программистском международном, то есть на английском), либо поручить эту работу системному программисту. Разумеется, не бесплатно. Но самостоятельное изучение режима ядра обойдется еще дороже. Эта не та область, которую можно постичь за месяц или два. В программировании драйверов есть множество неочевидных тонкостей, известных только профессионалам. Умение совмещать в себе проектирование баз данных с разработкой низкоуровневых компонентов даровано не каждому. Сосредоточься на чем-то одном - на том, что у тебя получается лучше всего. В противном случае ты впустую потратишь время, не получив ни денег, ни удовольствия.

Хакерский взгляд на Windows

В ЯДРЕ

"Вычислительная" часть (она же "движок") - сердце любого приложения, основной "интеллект" сосредоточен именно здесь. "Вычисления" - это не только математические расчеты, но и любая обработка данных вообще. В частности, вычислительная часть "Тетриса" трансформирует фигуры, подсчитывает очки, убирает заполненные строки, обрабатывает наложения спрайтов и т.д.

Чаще всего движок пишется на приплюснутом или классическом C, реже - на Фортране, Pascal'e и других экзотических языках. Выбор определяется личными пристрастиями программиста, с одной стороны, и возможностями языка - с другой. А язык - это не только компилятор, но еще и его окружение, то есть среда разработки, отладчик, верификаторы кода, библиотеки, системы обнаружения утечек памяти. Многие останавливаются на Visual Studio только из-за ее IDE. Интегрированный отладчик с перекомпиляцией на лету, автозавершение имен функций, удобные мастера. С непривычки все это здорово заводит и возбуждает, но, не дойдя до оргазма, эйфория угасает. Мастера тесно завязаны на MFC, а MFC использует нестандартные расширения и, вообще говоря, непереносим. Можно, конечно, писать и без мастеров, но что тогда остается от IDE? А текстовой редактор можно найти и покруче. Системы контроля и визуализаторы данных находятся в глубоко зачаточном состоянии. Основным лекарством становится отладчик, из-под которого не выползаешь ночами, но с ошибками синхронизации потоков он все равно не справляется. В мире *nix, где в основном используется "тяжелая" многозадачность, этих проблем просто не возникает, да и ассортимент инструментальных средств там побогаче. Есть мнение, что дешевле написать вычислительную часть на Linux и затем перекомпилировать под Windows, чем писать на Visual C++. Трудности разработки с лихвой компенсируются легкостью отладки. Правда, для этого следует проникнуться идеологией GDB - самого "правильного" отладчика в мире. Он совсем не похож на Turbo Debugger и намного более продвинут, чем SoftIce (правда, совершенно непригоден для взлома, но взлом - это другое дело).

Прежде чем опускать лапы на клавиатуру, мучительно соображая, что бы такое сейчас написать, обязательно обшарь все уголки Сети на предмет наличия уже готового кода. Программирование возникло не сегодня и не вчера. Все, что только можно было написать, уже написано! Допустим, тебе потребовалась своя версия кодека G.729 для создания мини-АТС или для организации телеконференции. Писать все с нуля? Мы что, рехнулись?! Открываем Google, выясняем, что стандартизацией этого протокола занимается комитет International Telecommunication Union, представленный сайтом www.itu.int, где можно заказать (правда, не бесплатно) не только описание самого алгоритма сжатия, но и исходные тексты кодеков с комментариями. Зная конкретно, что именно мы ищем, легко добываем файлы в Осле (правда, их там 600 метров с гаком, но Осел животное терпеливое, и не такие объемы перетаскивал). Как вариант, можно скачать библиотеку Intel IPP, в состав которой входит несколько версий кодека, оптимизированных под MMX и SSE. Помимо этого, в процессе поиска обнаруживается добрый десяток "студенческих" реализаций вполне приемлемого качества. Доступность исходных текстов большинства UNIX-приложений превращает программирование из творческой задачи в азартный поиск готовых кусков кода, которые порою обнаруживаются в самых неожиданных местах. Конечно, при этом всегда существует риск нарваться на чью-то ошибку (и тщательно замаскированную закладку в том числе), но... искушение всегда побеждает соблазн.

Разработчик движков - хороший алгоритмист и отчасти даже математик. Знания ассемблера и устройства операционной системы приветствуются, но в общем-то необязательны. Зато свой непосредственный язык программирования разработчик должен знать от и до, используя предоставляемые им возможности на полную катушку.

НА ПОВЕРХНОСТИ

Пользовательский интерфейс - это лицо и одежка программы, зачастую отнимающие более половины общего времени разработки проекта (а некоторые приложения, например, бух или склад, из одного интерфейса и состоят, "вычислительного" кода там очень немного). Интерфейс не обязательно должен быть графическим. Командная строка и консольный режим здравствуют и по сей день, однако сфера их применения уменьшилась до очень узкого круга (можно даже сказать "клуба") матерых профессионалов. Кворума мы не наберем, а выйти на массовый рынок без иконок нереально (вот она, скрытая религиозность!).

Эту интеллектуально-непритязательную, но трудоемкую работу целесообразнее всего поручить "пионерам" - начинающим программистам с дизайнерской жилкой, потому что разработчик интерфейсов в первую очередь художник, а уже после этого программист. Использование готовых пиктограмм не только безвкусно, но и пошло. Любая программа должна иметь свой собственный, легко узнаваемый "фирменный" стиль, выполненный в единой цветовой гамме и объединенный общей идеей. Стандартные ресурсы, входящие в комплект штатной поставки Visual Studio, ни на что не годятся (программы, написанные "для себя", мы в расчет не берем).

Интерфейс быстрее всего пишется на Delphi/C++ Builder/Visual Basic/Visual C++/.NET и прочих системах быстрой разработки приложений. Компактность кода и его быстродействие, конечно, оставляют желать лучшего, но кто на них обращает внимание? Главное - опередить конкурентов, не позволив им первыми выйти на рынок. Delphi, например, предпочтителен из-за того, что под него можно найти любые готовые компоненты на все случаи жизни. Позиция .NET, несмотря на связанную с ней масштабную маркетинговую политику, выглядит как-то неубедительно, и программисты все еще осторожничают с переходом. Почему? Главным нововведением в .NET стала виртуальная машина (.NET Framework) с промежуточной компиляцией приложений в P-код. Идея далеко не новая (даже на ZX-Spectrum, кажется, было что-то подобное).

Теоретически все выглядит блестяще - программист пишет программу на Visual Basic'e, Visual C++ или C# (клон Java), и она выполняется на любом процессоре и под любой осью, для которой эта виртуальная машина существует. Сними лапшу с ушей! Если есть прямые вызовы API-функций или управление оборудованием, о переносимости можно забыть. Если же их нет, исходный код, написанный в ANSI-стиле, транслируется любым ANSI-совместимым компилятором, без всякой виртуальной машины, кстати, съедающий львиную долю производительности. Под .NET существует несколько достойных библиотек для создания web-приложений, взаимодействия с серверами баз данных и т.д. и т.п., но до изобилия готовых компонентов, которыми славится Delphi, она явно не дотягивает. Возможно, со временем ситуация и изменится (если учесть рьяную озабоченность Microsoft, она изменится наверняка), но в настоящий момент .NET выигрывает лишь на тех задачах, на которые ее сориентировали, то есть она сильна в сетевых приложениях.

Вернемся к переносимости. Платформа .NET позиционируется как среда открытых стандартов. Язык виртуальной машины описывается документом ECMA-335 (его бесплатную pdf-версию можно скачать с www.ecma-international.org/publications/standards/Ecma-335.htm), а C# – ECMA-334 (www.ecma-international.org/publications/standards/Ecma-334.htm). Любой производитель может создавать собственную реализацию платформы .NET, не спрашивая у Microsoft разрешения и не выплачивая никаких отчислений.

Известны по меньшей мере два проекта переноса .NET в среду *nix. Первый, спонсируемый Free Software Foundation (сокращенно FSF), носит название DotGNU (www.dotgnu.org) и развивается в рамках одноименного проекта, частью которого является компилятор GCC. Второй проект называется Mono (www.go-mono.com) и спонсируется компанией Ximian, распространяется под лицензий GPL/LGPL и MIT License, что в ряде случаев намного предпочтительнее. Найти его можно в дистрибутиве Федоры. Знакомство с обоими проектами оставляет в душе лишь разочарование. Для реальной работы они непригодны. Из любви к искусству, конечно, можно и пострадать, истязая свою задницу, замученную бесконечным сидением у монитора, но программист с нормальной половой ориентацией не задумываясь выберет Delphi/Free Pascal или Visual C++ с MFC, бесплатные порты которой под *nix уже имеются.

А вот для Java-программистов открытый C# - это настоящая находка (что, собственно, и неудивительно, ведь его разработкой руководил Anders Hejlsberg. Да-да! Тот самый Anders Hejlsberg, который создал Delphi и Turbo-Pascal). Теперь судьба проекта не будет зависеть от правого мизинца левой пятки главы корпорации SUN! Впрочем, в программировании интерфейсов возможности и удобство языка глубоко вторичны и все опять-таки упирается в готовые библиотеки и компоненты. В этом смысле Java застряла между голым win32 API и MFC, то есть запрограммировать интерфейс на ней можно, но это же надо программировать - кодить, а не мышь по коврику гонять!

.NET-приложение под Mac OS

Крис Касперски aka мыщъх

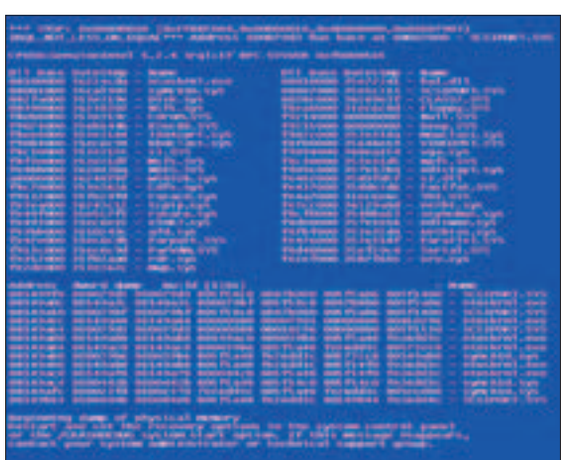
НЕПСИХОЛОГИЧЕСКИЕ ТЕСТЫ

РАЗБИРАЕМСЯ В ТЕСТИРОВАНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Программные ошибки коварны и злы. Сколько они загубили хороших проектов! Сколько времени было угрохано на поиски плавающих багов, затаившихся в засаде и выскакивающих во время демонстрации продукта заказчику. Но программист хитрее и терпеливее. Вооруженный современным инструментарием диагностики, он врезается в самое гнездилище багов и бьет их наповал.

В среднем тестирование отнимает 50% времени и 50% стоимости от общей сметы проекта (обязательно учитывай это закладывая бюджет). В больших компаниях (Intel, IBM, Microsoft) за каждым разработчиком закреплен тестер. Да, прошло то время, когда эту работу выполнял второсортный программист, которого еще не подпускали к самостоятельному кодированию (мол, прежде чем допускать свои ошибки, сначала пусть учатся на чужих). Сегодня тестер - это высококвалифицированный и хорошо оплачиваемый специалист, в услугах которого нуждаются тысячи фирм.

Когда тебе скажут, что жизненный цикл продукта состоит из проектирования, реализации, тестирования и поддержки, не верь! Тестирование сопровождает проект всю его жизнь - от момента рождения до самой смерти. Проектировщик закладывает механизмы самодиагностики и вывода "телеметрической" информации. Разработчик тестирует каждую написанную им функцию (тестирование на микроуровне). Бета-тестеры проверяют работоспособность всего продукта в целом. У каждого из них должен быть четкий план действий, в противном случае тестирование провалится еще не начавшись.



Случается и такое

ТЕСТИРОВАНИЕ НА МИКРОУРОВНЕ

В идеале для каждой функции исходного кода разрабатывается набор автоматизированных тестов, предназначенных для проверки ее работоспособности. Лучше всего поручить эту работу отдельной группе программистов, поставив перед ними задачу разработать такой пример, на котором функция провалится. Вот, например, функция сортировки. Простейший тест выглядит так. Генерируем произвольные данные, прогоняем через нее, и, если для каждого элемента N условие N <= N+1 (N >= N+1 для сортировки по убыванию) истинно, считаем, что тест пройден правильно. Но этот тест неправильный! Нужно убедиться, что на выходе функции присутствуют все исходные данные и нет ничего лишнего! Многие функции нормально сортируют десять или даже тысячу элементов, но спотыкаются на одном или на двух (обычно это происходит при сортировке методом деления). А если будет ноль сортируемых элементов? А если одна из вызываемых функций (например, malloc) возвратит ошибку, сможет ли тестируемая функция корректно обработать ее? Сколько времени (системных ресурсов) потребуется на сортировку максимально возможного числа элементов? Неоправданно низкая производительность - тоже ошибка!

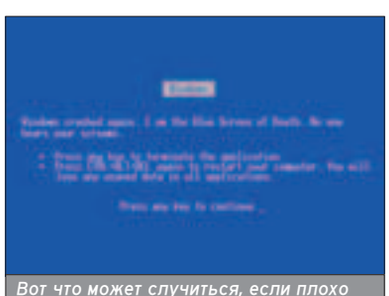
Существует два основных подхода к тестированию: черный и белый ящики. "Черный ящик" - это функция с закрытым кодом, проверка которой сводится к тупому перебору всех комбинаций аргументов. Очевидно, что подавляющее большинство функций невозможно протестировать за разумное время (количество комбинаций слишком велико). Код белого ящика известен, и тестер может сосредоточить свое внимание на пограничных областях. Допустим, в функции есть ограничение на предельно допустимую длину строки в MAX_LEN символов. Тогда следует тщательно исследовать строки в MAX_LEN-1, MAX_LEN и MAX_LEN+1 символов, поскольку ошибка "в плюс-минус один байт" - одна из самых распространенных.

Тест должен задействовать все ветви программы, чтобы после его выполнения не осталось ни одной незадействованной строчки кода. Отношение кода, который хотя бы раз получил выполнение, к общему коду программы называется покрытием (coverage), и для его измерения придумано множество инструментов - от профилировщиков, входящих в штатный комплект поставки компиляторов, до самостоятельных пакетов, лучшим из которых является NuMega True Coverage.

Разработка тестовых примеров - серьезная инженерная задача, часто даже более сложная, чем разработка самой "подопытной" функции. Неудивительно, что в реальной жизни к ней прибегают лишь в наиболее ответственных случаях. Функции с простой логикой тестируются "визуально", именно поэтому у нас все глючит и падает.

Всегда транслируй программу с максимальным уровнем предупреждений (для Microsoft Visual C++ это ключ /W4), обращая внимание на все сообщения компилятора. Некоторые наиболее очевидные ошибки обнаруживаются уже на этом этапе. Сторонние верификаторы кода (lint, smatch) еще мощнее и распознают ошибки, с которыми трансляторы уже не справляются.

Тестирование на микроуровне можно считать законченным тогда, когда функция компилируется несколькими компиляторами и работает под всеми



Вот что может случиться, если плохо тестировать программу

операционными системами, для которых она предназначена.

РЕГИСТРАЦИЯ ОШИБОК

Завалить программу - проще всего. Зафиксировать обстоятельства сбоя намного сложнее. Типичная ситуация: тестер прогоняет программу через серию тестов, непройденные тесты отправляются разработчику, чтобы тот локализовал ошибку и исправил баги. Но у разработчика эти же тесты проходят успешно! А... он уже все переделал, перекомпилировал с другими ключами и т.д. Чтобы этого не происходило, используй системы управления версиями - Microsoft Source Safe или *nix'овый CVS.

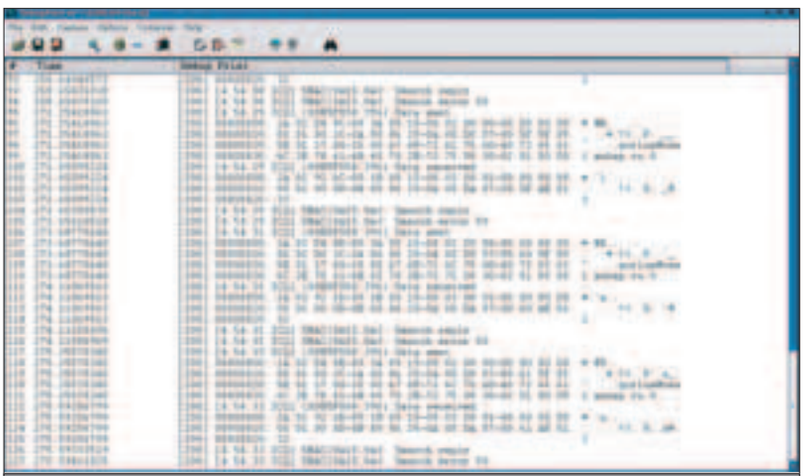
Сначала тестируется отладочный вариант программы, а затем точно так же финальный. Оптимизация - коварная штука, и дефекты могут появиться в самых неожиданных местах, особенно при работе с вещественной арифметикой. Иногда в этом виноват транслятор, но гораздо чаще - сам программист.

Самыми коварными являются "плавающие" ошибки, проявляющиеся с той или иной степенью вероятности: девятьсот прогонов программа проходит нормально, а после них неожиданно падает без всяких видимых причин. Многопоточные приложения и код, управляющий устройствами ввода/вывода, порождают особый класс невоспроизводимых ошибок, некоторые из которых могут проявляться лишь раз в несколько лет (!). Вот типичный пример:

```
char *s;
f1() {int x=strlen(s); s[x]='*'; s[x+1] = 0;}    // поток 1
f2() {printf("%s\n",s);}                          // поток 2
```

Один поток модифицирует строку, а другой выводит ее на экран. Программа будет работать нормально до тех пор, пока поток 1 не прервется в тот момент, когда звездочка уже уничтожила завершающий символ нуля, а новый ноль еще не был дописан. Легко доказать, что существуют такие аппаратные конфигурации, на которых эта ошибка не проявится никогда (для этого достаточно взять однопроцессорную машину, гарантированно успевающую выполнить весь код функции f1 за один квант). По закону подлости этой машиной обычно оказывается компьютер тестера, и у него все работает. А у пользователей - падает.

Чтобы локализовать ошибку, разработчику недостаточно знать, что "программа упала", необходимо сохранить и затем тщательно проанализировать ее состояние в момент обрушения. Как правило, для этого используется аварийный дамп памяти, создаваемый утилитами типа "Доктора Ватсона" (входит в штатный комплект поставки операционной системы) »



DebugView ловит чью-то отладочную информацию

ВЕРИФИКАТОРЫ КОДА ЯЗЫКОВ C\C++

Самый простой верификатор - это утилита lint, входящая в штатный комплект поставки большинства *nix-систем. Ее возможности сильно ограничены, а версия для Windows распространяется только на коммерческой основе.

Достойная альтернатива lint'у - открытый проект CLINT, распространяемый в исходных текстах, которые можно скачать с сервера сообщества "кузницы": **http://sourceforge.net/projects/clint**.

Еще мощнее SPLINT, нацеленный на автоматизированный поиск переполняющихся буферов и прочих программистских ошибок, которые не находят lint и CLINT. Это серьезный, хорошо документированный продукт, распространяющийся в исходных текстах на некоммерческой основе, уже скомпилированный под Windows, Linux, Solaris и FreeBSD (CLINT поставляется только в исходных текстах, с которыми еще предстоит повозится). Лежит здесь: **http://lclint.cs.virginia.edu**.

Smatch C source checker представляет собой автоматический анализатор исходного кода для нахождения типовых ошибок (утечек памяти, переполнений буфера, паразитных NULL-указателей и т.д.), созданный в рамках проекта по выявлению ошибок в Linux-ядре. Распространяется в виде патчей к gcc-компилятору и набора perl-скриптов для анализа дампов. И снова это проект "кузницы": **http://smatch.sourceforge.net**.

Совершенно иной подход исповедует MLC, он же Meta-Level Compilation (компилятор метауровня), транслирующий программу в промежуточный код и за счет доступа к абстрактному синтаксическому дереву, обнаруживающий трудноуловимые ошибки, пропущенные остальными верификаторами. Разработчики утверждают, что с помощью метакомпилятора им удалось выявить свыше 500 ошибок в реально существующих системах, таких как Linux, OpenBSD, Xok, Stanford FLASH и gp. В настоящее время MLC распространяется в виде бесплатного компилятора xgcc, базирующегося на GNU C, и вспомогательного транслятора metal для создания расширений. Найдется тут: **http://metacomp.stanford.edu**.

Столкнувшись с необъяснимой ошибкой, начинающие программисты обычно обвиняют компилятор, хотя в подавляющем большинстве случаев они виноваты сами. Невнимательное чтение документации и небрежный стиль кодирования - основные враги программиста.

clint вместе с исходниками можно скачать с sourceforge.net

или, на худой конец, значение регистров процессора и содержимое стека. Поскольку не все ошибки приводят к аварийному завершению программы, разработчик должен заблаговременно предусмотреть возможность создания дампов самостоятельно - по нажатию специальной комбинации клавиш или при срабатывании внутренней системы контроля.

БЕТА-ТЕСТИРОВАНИЕ

Собрав все протестированные модули воедино, получаем минимально работоспособный продукт. Если он запускается и не падает - это уже хорошо. Говорят, посади за компьютер неграмотного человека, и пусть он давит на все клавиши, пока программа не упадет. Ну да как же! Тестирование программы - это серьезная операция, и такой пионерский подход здесь неуместен. Необходимо проверить каждое действие, каждый пункт меню, на всех типах данных и операций. Бета-тестер может и не быть программистом, но он обязан иметь квалификацию продвинутого пользователя.

Как разобраться в файле дампа? Нужно знать язык ассемблера! Без этого никогда не стать настоящим профессионалом!

Уронив программу (или добившись от нее выдачи неверных данных), бета-тестер должен суметь воспроизвести сбой, то есть выявить наиболее короткую последовательность операций, приводящую к ошибке. А сделать это ой как непросто! Попробуй-ка вспомнить, какие клавиши были нажаты! Что? Не получается?! Су. Используй клавиатурные шпионы. На любом хакерском сайте их валом. Пусть поработают на благо народа (не вечно же пароли похищать). Шпионить за мышью намного сложнее: приходится сохранять не только позицию курсора, но и координаты всех окон или задействовать встроенные макросредства (по типу Visual Basic'a в Word). В общем, мышь, по-моему - это саксь и маст дай. Нормальные бета-тестеры обходятся одной клавиатурой. Полный протокол нажатий сужает круг поиска ошибки, однако с первого раза воспроизвести сбой удается не всегда и не всем.

www.testingcraft.com - проект умер, но все еще есть куча полезных ссылок по тестированию.

В процессе тестирования приходится многократно выполнять одни и те же операции. Это раздражает, это ненадежно и непроизводительно. В штатную поставку Windows 3.x входил клавиатурный проигрыватель, позволяющий автоматизировать такие операции. Теперь же его приходится приобретать отдельно. Впрочем, такую утилиту можно написать и самостоятельно, в чем тебе помогут функции FindWindow и SendMessage.

Тестируй программу на всей линейке операционных систем: Windows 98, Windows 2000, Windows 2003 и т.д. Различия между некоторыми очень значительны. Что стабильно работает под одной осью, может падать под другой, особенно если она перегружена кучей конфликтующих приложений. Хорошо если это кривая программа Васи Пупкина (тут на пользователя можно и наехать), но если твоя программа не уживается с MS Office или другими продуктами крупных фирм, бить будут тебя. Никогда не меняй конфигурацию системы в процессе тестирования! Тогда будет трудно установить, чей это баг. Хорошая штука - виртуальные машины (VM Ware, Microsoft Virtual PC). На одном компьютере можно держать множество версий операционных систем с различной комбинацией установленных приложений - от стерильной до полностью захламленной. При возникновении ошибки состояние системы легко сохранить на жесткий диск и обратиться к нему впоследствии столько раз, сколько потребуется.

У тебя программа работает, а у пользователя - нет. Что делать?! Для начала - собрать информацию о конфигурации его компьютера ("Панель управления"->"Администрирование"->"Управление компьютером"->"Сведения о системе"). К сожалению, установить виновника таким путем сходу не удастся. Может, там вирус сидит и вредительствует. Или глючит какой-нибудь драйвер. Но, имея несколько отчетов от различных пользователей, в них можно выявить некоторую закономерность. Например, программа не идет на таком-то процессоре или на такой-то видеокарте.

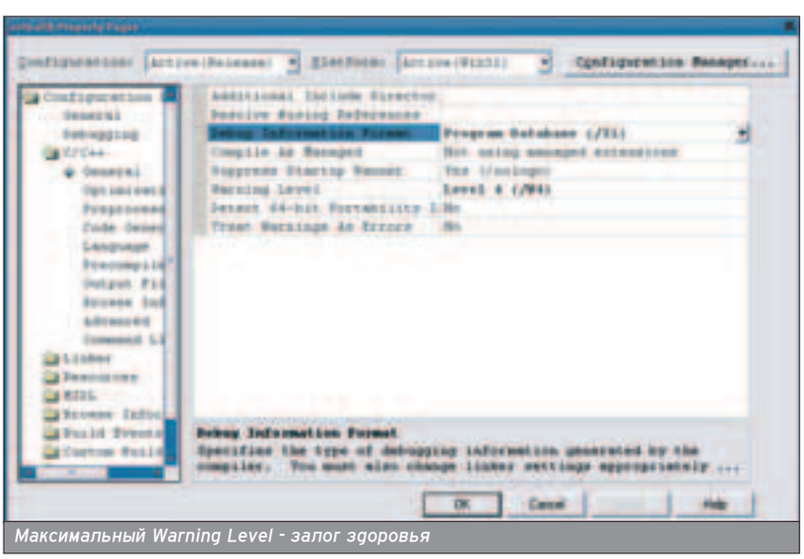
Другая возможная причина - утечка ресурсов. Утечки возникают всякий раз, когда программа злостно не освобождает то, что постоянно запрашивает. Чаще всего приходится сталкиваться с утечками памяти, но ничуть не хуже утекают перья, кисти, файловые дескрипторы и т.п. В общем, практически любые объекты ядра, USER и GDI. Тестер, работая с программой непродолжительные отрезки времени, может этого и не заметить (особенно если у него стоит Windows NT/2000/XP, в которой ресурсы практически неограниченны), но при "живой" эксплуатации у пользователей появляются огромные проблемы. Сначала легкое замедление быстродействия системы, затем конкретные тормоза, переходящие в полный завис, и, наконец, reset, сопровождаемый колоритным матом.

Отладочные библиотеки, входящие в состав компилятора Microsoft Visual C++, легко обнаруживают большинство утечек памяти. В сложных случаях приходится прибегать к верификаторам кода или динамическим анализаторам наподобие NuMega Bounds Checker. Но высшей инстанцией является эксперимент. Запусти "Диспетчер Задач Windows NT" и некоторое время поработай с тестируемой программой. Вкладка "Процессы" отображает текущие счетчики дескрипторов, размер выделенной памяти и т.д. (по умолчанию видны лишь некоторые из них, зайди в меню "Вид"->"Выбрать столбцы" и взведи все галочки). Если какой-то счетчик неуклонно увеличивает свое значение после некоторых операций - это утечка.

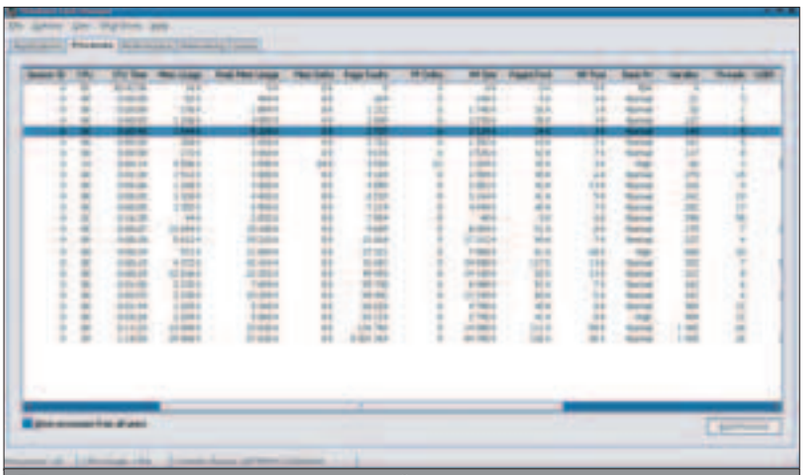
Для исследования работоспособности программы в условиях катастрофической нехватки ресурсов (памяти, дискового пространства) Microsoft включила в состав Platform SDK утилиту Stress.exe, снабдив ее иконкой танцующего мамонта. Корректно спроектированное приложение должно выживать при любых обстоятельствах. Обломали с выделением памяти из кучи? Переходи на резервный источник (стек, секция данных). Освободи все ненужное, но любой ценой сохрани все данные! Всегда сохраняй при старте программы минимально необходимое количество памяти "про запас", а потом используй его как НЗ. То же самое относится и к дисковому пространству.

ВЫВОД ДИАГНОСТИЧЕСКОЙ ИНФОРМАЦИИ

Самое страшное - когда программа неожиданно делает из обрабатываемых чисел "винегрет". Совершенно непонятно, кто в этом виноват и откуда надо плясать. Ошибка в одной функции может аукаться в совершен-



Максимальный Warning Level - залог здоровья



Диспетчер задач выводит массу интересной информации

Забавно, но многие фирмы предпочитают "документировать" ошибки, экономя на их исправлении.

но посторонних и никак не связанных с ней местах. Удар по памяти, искажение глобальных переменных или флагов (со)процессора. Здесь дамп уже не помогает. Застывшая картина статичного слепка памяти не объясняет, с чего началось искажение данных, в каком месте и в какое время оно произошло.

Для локализации таких ошибок в программу заблаговременно внедряются "телеметрические" механизмы для генерации диагностической информации. В идеале следовало бы протоколировать все действия, выполняемые программой, запоминая все машинные команды в специальном буфере. Собственно говоря, SoftIce в режиме обратной трассировки (back trace) именно так и поступает, позволяя нам прокручивать программу задом наперед. Это чудовищно упрощает отладку, но... как же оно тормозит! Искусство диагностики как раз и состоит в том, чтобы отобрать минимум важнейших параметров, фиксирующих максимум происходящих событий. По крайней мере, отмечай последовательность выполняемых функций вместе с аргументами.

Чаще всего для этой цели используется тривиальный fprintf для записи в файл или syslog для записи в системный журнал (в Windows NT это осуществляется вызовом API-функции ReportEven, экспортируемой библиотекой ADVAPI32.DLL). Начинающие допускают грубую ошибку, включая диагностику только в отладочную версию и удаляя ее из финальной:

```
#ifdef _DEBUG_
fprintf(flog, "%s:%d a = %08Xh; b = %08Xh\n",
__FILE__,__LINE__,a,b);
#endif
```

Когда такая программа упадет у пользователя, в руках программиста не окажется никакой диагностической информации, которая могла бы дать хоть какую-то зацепку. Лучше поступать так:

```
if (_DEBUG_)
 fprintf(flog, "%s:%d a = %08Xh; b = %08Xh\n",
__FILE__,__LINE__,a,b);
```

Если сбой повторяется регулярно, пользователь сможет взвести флажок DEBUG в настройках программы, и в следующий раз, когда она упадет, передаст программисту диагностический протокол, если, конечно, не переметнется к конкурентам. Штука.

Правильный вариант выглядит так:

```
if (2*2 == 4)
fprintf(flog, "%s:%d a = %08Xh; b = %08Xh\n",
__FILE__,__LINE__,a,b);
```

Грамотно отобранная телеметрическая информация отнимает совсем немного места и должна протоколироваться всегда (естественно, за размером log-файла необходимо тщательно следить, лучше всего если он будет организован по принципу кольцевого буфера). Некоторые программисты используют функцию OutputDebugString, посылающую информацию на отладчик. Если отладчик не установлен, можно воспользоваться утилитой Марка Руссиновича DebugView или аналогичной ей. Впрочем, пользы от такого решения все равно немного. Это тот же логинг, включаемый по требованию, а логинг должен быть включен всегда!

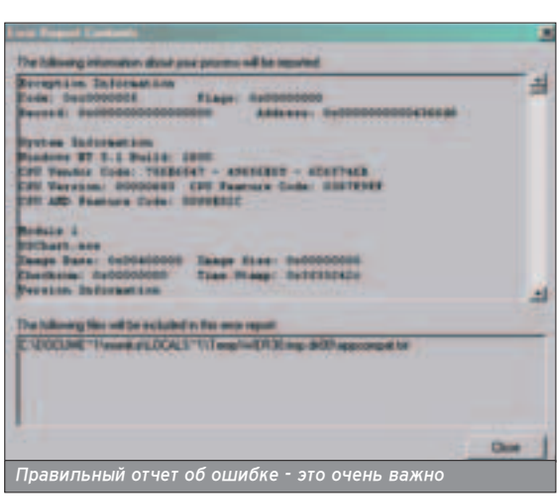
Основной недостаток fprintf в том, что при аварийном завершении программы часть телеметрической информации необратимо теряется (буфера остались несброшенными). Если же их сбрасывать постоянно, скорость выполнения программы ощутимо замедляется. Записывай телеметрию в разделяемую область памяти, а при возникновении сбоя сохраняй в файл протокола из параллельного процесса. Так будут и волки сыты, и овцы целы.

ЗАКЛЮЧЕНИЕ

Поиск ошибок не прекращается никогда! Даже когда продукт умирает, какие-то его компоненты используются в следующих версиях и в них вылезают новые баги. Ошибки так же неисчерпаемы, как и атом! Они образуют толстые многолетние наслоения, из-за которых руки чешутся и просятся в бой с ними, но начальство строго-настрого запрещает трогать. Это можно сравнить с притиркой механизма. По мере своего взросления модули все дальше и дальше уходят от первоначальных спецификаций. Теперь, чтобы написать совместимую функцию, необходимо тщательно проанализировать исходный код "старушки", постоянно ломая голову над вопросами "это баг или так задумано?" Основное правило разработчика гласит: не трогай того, что и так работает.

Забавно, но многие фирмы предпочитают "документировать" ошибки, экономя на их исправлении. В Базе Знаний или руководстве пользователя авторитетно заявляется: "Туда ходить не надо, кто не послушался - сам виноват". Возможности, которые так и не удалось отладить, но которые нельзя заблокировать или изъять, просто не документируются. Все ими пользуются (еще бы! самая "вкусность" продукта сосредоточена именно здесь), у всех все падает, но никто не может предъявить претензию, потому что никому ничего и не обещали.

Так что тестирование программного обеспечения - это не только инженерия, но еще политика и маркетинг. Выживает не тот, чей продукт лучше, а тот, кто правильно его "позиционирует". В конечном счете, любую ошибку можно превратить в достоинство.



Правильный отчет об ошибке - это очень важно

Филипп Коряка (phil@pereslavl.ru)

ТЕЛЕФОННОЕ ПРОГРАММИРОВАНИЕ

С ЧЕМ ЕДЯТ J2ME, ИЛИ КАК ЗАПРОГРАММИРОВАТЬ ТЕЛЕФОН

На современном рынке мобильных телефонов модели, которые умеют только исполнять свои прямые обязанности, то есть звонить, активно вытесняются более новыми - с поддержкой технологии J2ME. Будем разбираться, что это такое и как оно может пригодиться программисту.

Вынужден с прискорбием сообщить тебе, что мы живем в очень сложное время, когда все уже давно изобретено до нас, а новые чудеса техники творят только коллективы изобретателей. И только гений, причем не любой, способен изобрести что-то в одиночку. Все сказанное, конечно же, относится и к программистам, потому что они, по сути, те же изобретатели, просто их изобретения всегда похожи друг на друга. Эти изобретения - программы. Сейчас довольно сложно представить себе софт для ПК, который можно было бы написать в одиночку за разумное время, а потом еще и продавать всем желающим. Если только это не будет действительно что-то маленькое и эксклюзивное, до чего никто другой до сих пор не додумался. Что же делать среднестатистическому программисту, которого до сих пор не посетила эксклюзивная мысль? Тут есть несколько вариантов. Первый и самый простой - устроиться в компанию, занимающуюся разработкой софта, совместно с другими ее сотрудниками заниматься творчеством и получать за это деньги.

Но поскольку писать в журналах о простых и тривиальных путях не принято, я, пожалуй, расскажу о втором пути, более сложном, но в то же время более интересном. Его суть - найти некоторую нишу для программиста, в условиях которой он будет на равных с другими ее участниками. Это может быть, например, программирование для мобильных устройств, которым в наше время занимается гораздо меньше компаний, чем программированием для ПК. Многие вещи в этой области сейчас находятся на стадии зарождения. Если Фортуна начнет улыбаться тебе, ты сможешь не только заработать денег, но и получишь все шансы захватить пальму первенства в этой нише и, может быть, даже стать "Биллом Гейтсом мира мобильных телефонов" :).

ЧТО ТАКОЕ J2ME

Существуют различные технологии создания софта для мобильных устройств, или, как его еще принято называть, мидлетов, а главным предметом статьи станет технология J2ME (Java 2 Micro Edition), которая была специально разработана для устройств с ограниченными ресурсами памяти и вычислительной мощности, такими как сотовые телефоны.

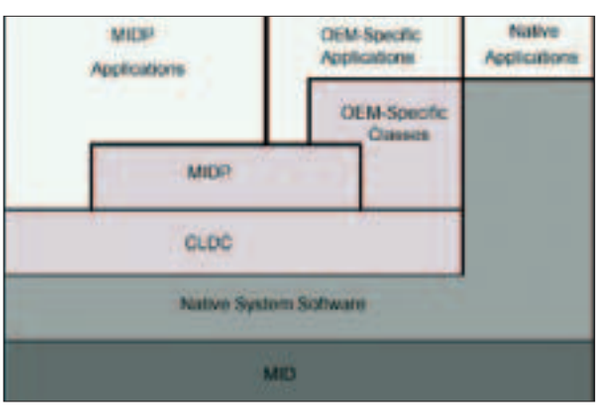
В этой статье мы будем в первую очередь говорить о программировании для сотовых телефонов, но J2ME - это гораздо более объемное понятие. J2ME включает в себя несколько конфигураций, каждая из которых определяет среду выполнения J2ME-приложения, то есть указывает, какая виртуальная машина используется, какие классы доступны и т.д. Для конфигурации могут быть определены так называемые профили, вносящие некоторые корректировки в среду исполнения. Например, профиль может корректировать набор доступных классов. Далее подробно расскажу о конфигурации CLDC (Connected, Limited Device Configuration) и о профиле MIDP (Mobile Information Device Profile).

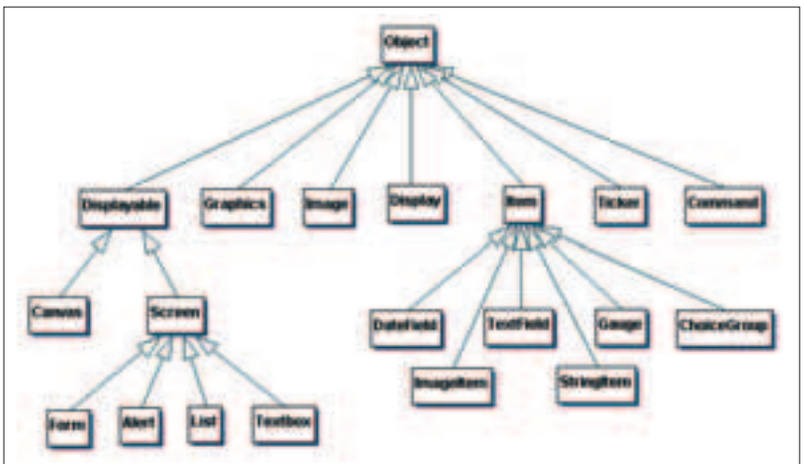
КОНФИГУРАЦИЯ CLDC

Не буду полностью описывать конфигурацию CLDC, остановлюсь на некоторых основных моментах. В первую очередь это, конечно, виртуальная машина. Согласно конфигурации CLDC используется виртуальная машина KVM, которая имеет ряд ограничений по сравнению, скажем, с JVM, однако это позволило разработчикам сделать ее очень компактной. Благодаря размеру в несколько сот килобайт KVM умещается в мобильные телефоны даже с очень небольшим количеством памяти.

Увы, и здесь не обошлось без недостатков.

- Отсутствие операций с плавающей точкой из-за отсутствия поддержки таких операций в мобильных устройствах на аппаратном уровне.
- Невозможность использования финализаторов из-за необходимости упрощения механизма сборки мусора.
- Ограниченность механизма обработки исключений, вызванная сложностью его реализации в полном объеме на мобильных устройствах.





Иерархия классов в J2ME

Классы, доступные в рамках CLDC, можно разделить на два множества. Первое - классы, унаследованные от J2SE (java.lang.*, java.util.*, java.io.*), второе - специфичные классы javax.microedition.*.

ПРОФИЛЬ MIDP

MIDP расширяет конфигурацию CLDC, добавляя к основному набору классов специфичные. Тут и классы для работы с пользовательскими интерфейсами, и реализующие механизмы хранения, и многие другие. Сегодня есть две версии профиля MIDP - это MIDP 1.0 и MIDP 2.0. Пока на рынке телефонов с первой версией большинство, но прогресс не стоит на месте и с каждым днем множит телефоны, соответствующие MIDP 2.0.

На мой взгляд и взгляд еще нескольких тысяч программистов, MIDP 1.0 не позволяет сделать ничего серьезного. Разработчики сотовых телефонов давно были единодушны с нами, поэтому им приходилось вводить для своих продуктов специфичный набор классов, расширяющий возможности программирования под конкретную модель. В результате нарушался основной принцип языка Java - платформонезависимость. Мидлеты, написанные, скажем, с использованием специфичных классов для телефонов Siemens, не работали на телефонах Nokia и наоборот. Позднее наступило прозрение, производители телефонов собрались вместе и придумали MIDP 2.0, в который включили много полезных классов, отсутствовавших в MIDP 1.0. Вот лишь некоторые из отличий MIDP 2.0:

- усовершенствованный пользовательский интерфейс;
- поддержка технологии Push Registry, согласно которой возможна активация мидлетов при получении мобильным устройством информации;
- полноценная реализация функций для работы со звуком;
- игровой API.

Конечно, и MIDP 2.0 не решает всех проблем, но это уже гораздо лучше, чем MIDP 1.0.

СТАВИМ СОФТ

Хватит теории, пора переходить к практике. Какой софт должен быть установлен на машине начинающего J2ME-программера? Конечно же, тут есть масса вариантов, но мы остановимся лишь на одном из них, на мой взгляд, самом простом и удобном. Итак, в первую очередь нам, конечно же, понадобится J2SE SDK. Куда без него? Далее необходимо установить инструментарий J2ME Wireless Toolkit (WTK), в котором есть почти все необходимое для создания мидлетов. В принципе установка этих двух средств не вызывает сложностей. Единственное, что могу посоветовать - не использовать пробелы в пути к папкам, в которые ты будешь устанавливать J2SE и WTK: это может сильно усложнить и без того непростую жизнь программера.

В принципе всего установленного вполне достаточно для работы. Но очень скоро ты столкнешься со следующей проблемой. Написав мидлет и отладив его в WTK, ты с удивлением обнаружишь, что он не работает в реальном телефоне или работает, но неправильно. Дело в том, что в состав WTK входит несколько эмуляторов телефонов, но при использовании любого из них нельзя быть уверенным, что твой мидлет заработает на той или иной модели телефона. Поэтому следует установить эмулятор именно той модели, »

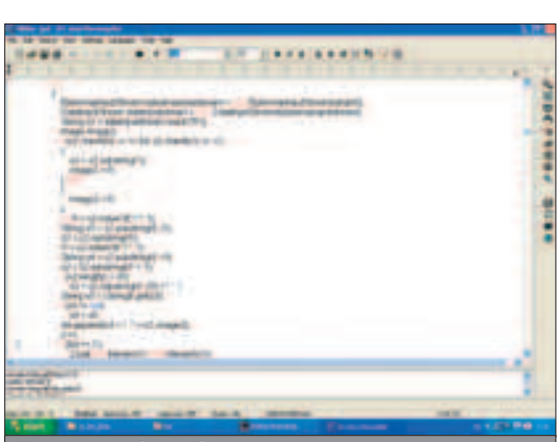
СТРАШНОЕ СЛОВО «ОБФУСКАТОР»

class-файлы, полученные при компиляции мидлета, содержат названия всех переменных, методов и классов, содержащихся в исходниках мидлета. Таким образом, декомпилировав их, можно получить практически первоначальный код. Очевидно, что вместе с jar-файлом своего мидлета ты фактически распространяешь его исходники. Чтобы решить эту проблему, были придуманы специальные программы - обфускаторы.

Обфускатор заменяет имена всех переменных, классов и методов на маловразумительные, но правильные с точки зрения синтаксиса языка Java сочетания символов. Кроме того, обфускаторы могут запутывать код программы при помощи различных хитрых приемов. В результате программа, подвергнутая обфускации, становится практически нечитаемой для человека, но синтаксически правильной для компилятора. Для более высокого уровня защиты программ можно поочередно использовать несколько обфускаторов, при этом каждый будет добавлять в код свою лепту хаоса.

Помимо функции защиты исходных кодов, обфускаторы обладают еще одним неоспоримым плюсом: размер class-файлов, полученных при помощи обфускаторов, как правило, меньше из-за использования коротких имен. Как известно, для мобильных устройств размер имеет очень большое значение.

Дополнительную информацию об обфускаторах для J2ME сможешь найти по адресу: **http://developers.sun.com/prodtech/javatools/jsstandard/reference/techart/obfuscation.html**.



Результат работы обфускатора

для которой пишется программа, и произвести отладку на нем. Как правило, все приличные разработчики телефонов (а зачем нам писать софт для неприличных разработчиков?) дают скачать со своих сайтов бесплатные эмуляторы.

ЗАГРУЖАЕМ МИДЛЕТЫ В ТЕЛЕФОН

Рано или поздно в жизни любого J2ME-программиста наступает момент, когда ему надоедает отлаживаться в эмуляторе и у него возникает непреодолимое желание загрузить в телефон написанный мидлет. Сделать это можно несколькими способами в зависимости от выбранного способа связи компьютера и телефона. Самый сложный способ - соединить телефон с компьютером data-кабелем. Тут потребуется некоторый менеджер закачек, подобранный под определенную модель телефона.

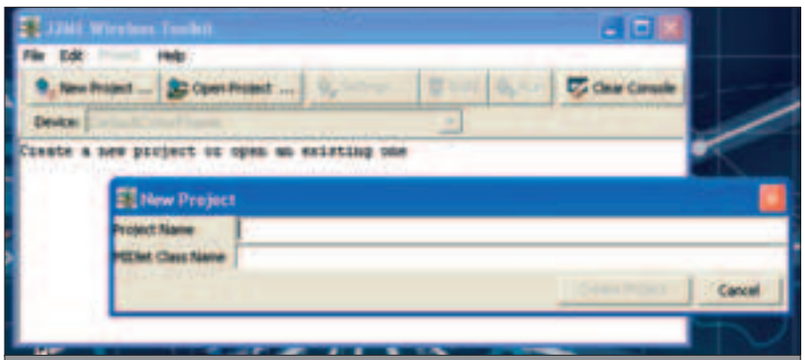
Следующими по сложности идут загрузка мидлета через WAP или HTTP. В этом случае потребуется хостинг, на котором можно будет разместить мидлет, и настройка GPRS в телефоне. Кроме того, этот способ требует некоторых скромных капиталовложений.

Самыми простыми, конечно же, являются загрузка через IrDA (инфракрасный порт) и Bluetooth. Тут, скорее всего, будет достаточно софта, уже установленного на твоем ПК. При условии, конечно, что ПК имеет IrDA или BT. Вот, пожалуй, и все известные мне способы, а на каком остановиться - решай сам.

СОЗДАЕМ ПРОЕКТ

Теперь, когда ты установил весь необходимый софт, можно переходить непосредственно к созданию мидлета. Заходи в директорию с WTK и запускай там KToolbar.exe. Жми кнопку New Project и указывай имя проекта и название класса мидлета. Вообще говоря, проект может содержать несколько мидлетов, а дополнительные можно будет добавить позже.

После нажатия кнопки Create Project твоему взору предстанет окно с установками для мидлета. В принципе сейчас их можно проигнорировать, а позже изменить или из меню KToolbar, или вручную отредактировав JAD-файл.



Создание нового проекта

Цены на рынке J2ME-софта, как правило, очень небольшие, однако это компенсируется высокими объемами продаж.

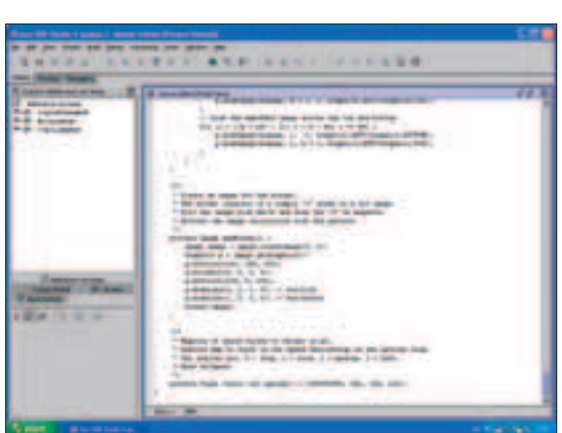
После создания проекта в директории WTK\apps появится поддиректория с названием, совпадающим с именем созданного проекта. Она имеет следующую структуру:

- \bin - сгенерированные jad и jar файлы;
- \classes - откомпилированные и верифицированные классы;
- \lib - сторонние используемые библиотеки, например, классы, специфичные для какой-либо модели телефона;
- \res - различные ресурсы: иконки, тексты и т.д.;
- \src - исходники всех мидлетов проекта;
- \tmpclasses - откомпилированные, но не верифицированные классы;
- \tmplib - используется совместно с \lib.

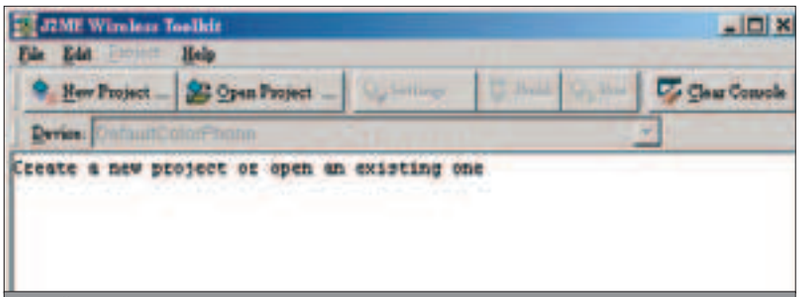
ПИШЕМ ПЕРВЫЙ МИДЛЕТ

Повторю, что исходные коды мидлета находятся в директории WTK\apps\your_project_name\src\. С помощью любого редактора текстов нужно создать в этой папке файл с именем, совпадающим с именем мидлета. Если ты уже забыл название своего мидлета, просто нажми в KToolbar кнопку Build и жди сообщения о том, что не удается найти файл "твой_мидлет.java". Это и есть заветное имя. Вот текст HelloWorld-мидлета и мои комментарии к нему:

```
// Импортируем классы J2ME
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class HelloMidlet extends MIDlet
implements CommandListener{
// Дисплей для твоего мидлета
```


Среда разработки J2ME-приложений


Эмуляторы от Siemens



KToolbar

```
private Display midletDisplay;

// Команда для выхода из мидлета
private Command exitCommand;

public HelloMidlet(){
 // Получаем доступ к дисплею
 midletDisplay = Display.getDisplay(this);

 // Инициализируем команду exitCommand
 exitCommand = new Command("Exit", Command.SCREEN,
1);
}

// Этот метод вызывается при запуске мидлета
public void startApp(){
 // Создаем текстовое поле с заголовком "Hello
Midlet" и текстом "Hello World!"
 TextBox textBox = new TextBox("Hello Midlet", "Hello
World!", 256, 0);

 // Добавляем ранее созданную команду exitCommand
к текстовому полю
 textBox.addCommand(exitCommand);

 // Устанавливаем обработчик комманд
 textBox.setCommandListener( (CommandListener) this);

 // Отображаем на дисплее текстовое поле
 midletDisplay.setCurrent(textBox);
}

// Этот метод вызывается в случае приостановки вы-
полнения мидлета
public void pauseApp(){

}

// Этот метод вызывается при завершении работы
мидлета
public void destroyApp(boolean unconditional){

}

// Тут происходит обработка команд
public void commandAction(Command command,
Displayable screen){
 // Если команда exitCommand
 if (command == exitCommand){
 // Вызываем destroyApp
 destroyApp(false);

 // Сообщаем платформе о завершении мидлета
 notifyDestroyed();
}
}
}
```

После запуска этого мидлета на экране телефона появится текстовое поле, содержащее текст "Hello World!". Также покажется команда Exit, выбрав которую можно выйти из мидлета. Пример, конечно, тривиальный, но он демонстрирует общий принцип построения мидлетов.

КОМПИЛИРУЕМ МИДЛЕТ

Все не так уж и сложно: в KToolbar жмем кнопку Build, переживаем одно мгновение (ну или не совсем мгновение, если исходники твоего мидлета уже занимают несколько сотен килобайт) и с радостью встречаем скомпилированный мидлет. После этого можно будет приступать к сборке приложения, для чего выполняется команда Project->Package->Create Package. Результатом будут jar- и jad-файлы в директории bin, которые можно смело закачивать в телефон или запускать в эмуляторе.

jar-файл представляет собой архив с классами твоего мидлета. Кстати, его можно разархивировать обычным zip'ом.

jad-файл является обычным текстовым файлом и содержит информацию о мидлете. В нем, например, содержится название мидлета, имя производителя, указание версии и т.д. Иногда там содержится более полезная информация. Например, при использовании Push Registry там можно найти сведения о том, с какими данными следует проассоциировать данный мидлет.

ЗАПУСКАЕМ МИДЛЕТ

Запустить мидлет можно непосредственно из Ktoolbar кнопкой Run. После этого мидлет будет запущен в одном из доступных стандартных эмуляторов:

- MinimumPhone - совсем простой телефон;
- DefaultGrayPhone - стандартный телефон с монохромным экраном;
- DefaultColorPhone - стандартный телефон с цветным экраном;
- Pager - пейджер.

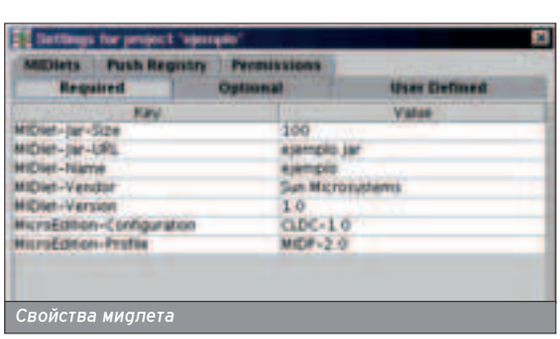
Как я уже говорил, стандартные эмуляторы не всегда подходят для отладки, поэтому ты можешь использовать специализированные эмуляторы конкретных моделей телефонов. В них, как правило, нет ничего сложного. После запуска эмулятора выбираешь в меню пункт "Запустить приложение" и в появившемся окне указываешь jad-файл твоего мидлета. Кроме того, имеется возможность настроить KToolbar на использование нестандартного эмулятора.

Эмулятор

ПРОДАЕТСЯ ЛИ J2ME-СОФТ?

Теперь, когда ты приобщился к сообществу J2ME-программистов, самое время задаться вопросом о том, как продать мидлет? Варианты тут, в принципе, те же, что и с софтом для ПК. Можно попробовать делать это самостоятельно через интернет, что чревато всеми трудностями продвижения shareware. А можно обратиться к одной из компаний, занимающихся продажей мелодий, картинок и софта для мобильников. Единственная неприятность при этом - обязанность отдавать некоторый процент от продаж представляющей тебя компании.

В то же время твой продукт будут рекламировать бесплатно. Кроме того, заключая контракт на продажу мидлета, ты, как правило, получаешь в нагрузку тестеров, которые проверяют твою программу на различных телефонах, так что процент от продаж вполне окупается. Для заключения контракта от тебя не требуется ничего сверхъестественного. Достаточно просто выбрать подходящую компанию, благо таких сейчас много, и написать им письмо с предложением продавать твой мидлет. Цены на рынке J2ME-софта, как правило, очень небольшие, однако это компенсируется высокими объемами продаж. Еще одним неоспоримым плюсом рынка J2ME является почти полное отсутствие пиратства в отличие от рынка софта для ПК. Так что дерзай. Возможно, J2ME - это именно то, что поможет тебе разбогатеть.



Свойства мидлета

Алексей Башкеев (botan@dezcom.mephi.ru)

ПО ТУ СТОРОНУ КОДИНГА

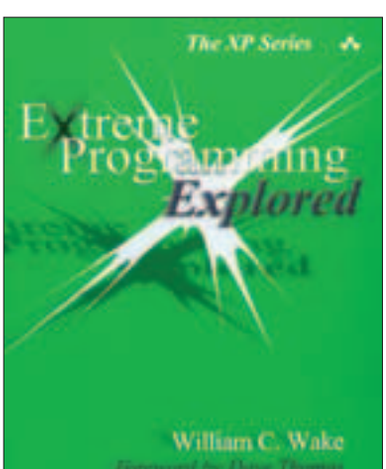
ОБЗОР СУЩЕСТВУЮЩИХ ТЕХНОЛОГИЙ ПРОГРАММИРОВАНИЯ И ОБЛАСТЕЙ ИХ ПРИМЕНЕНИЯ

В этой статье ты узнаешь о той части процесса программирования, которая стоит за процессами "написания кода". В этом мире свои законы, правила и технологии, практически одинаковые для всех средств программирования.

Ты уже умеешь программировать и написал не один десяток программ. Из этой статьи ты узнаешь, как можно оптимизировать этот процесс, как достигнуть лучших результатов при минимальных усилиях. С чего начать реализацию большого проекта? Как автоматизировать поиск ошибок в программах? Что делать, если ты уже выпустил одну версию, разрабатываешь следующую, а в выпущенной версии обнаружилась ошибка? Как организуется совместная работа нескольких программистов над одним и тем же проектом? На чем бы ты ни писал, ты столкнешься со всеми этими проблемами. Могу тебя утешить - ты не первый, кто с ними сталкивается. Такие вопросы встают и перед огромными корпорациями, и перед разработчиками бесплатного софта любой страны мира. Для их решения уже существуют средства - от программ до целых философий.

ЭКСТРЕМАЛЬНОЕ ПРОГРАММИРОВАНИЕ (ХП)

Если ты никогда о нем не слышал, то, скорее всего, подумаешь о "сжатых сроках" программирования - по 20 часов в сутки или же о программировании в ситуациях, в которых высока цена ошибки и на плечи программиста ложится громадная ответственность. Один мой знакомый до сих пор считает, что ХП - это программирование в условиях Крайнего Севера без отопления. Спешу тебя огорчить, ХП - это лишь "небольшой набор конкретных правил, позволяющих максимально эффективно (читай: без напряжения и непроизводительной деятельности) выполнять требования современной теории управления программными проектами". Философия ХП основана на анализе многих успешных и провальных проектов. В результате такого анализа и родился вышеупомянутый набор правил. И хотя многие правила рассчитаны на крупные проекты с большим числом разработчиков, некоторые из них могут быть использованы и по отдельности. Полный обзор всех методов ХП можно найти на русском сайте www.xprogramming.ru, а я лишь коротко расскажу о некоторых из них.



По экстремальному программированию написаны сотни книг

USER STORY

С чего начинается большой проект? С постановки технического задания, скажешь ты и будешь прав. Но далеко не всегда заказчик является грамотным человеком и точно знает, что ему надо. Бывает, что заказчиков несколько и каждый видит конечный результат "немного по-другому". Заказчики вообще кажутся разработчикам довольно странными людьми и наоборот. В ХП эту проблему "недопонимания" решают следующим образом: заказчику предлагается написать User Stories (карточки пользователя). В них заказчик описывает элементы интерфейса и функциональности будущей системы на простом человеческом языке, что должно быть положено в основу технического задания и на основе которых можно выяснить, достигнут ли конечный результат. В общем, очень полезный инструмент для формирования технического задания.

User Story выглядит страшновато, но в работе помогает

О ДИЗАЙНЕ И ИНТЕРФЕЙСЕ В ХП

"Реализовывать не то, что может пригодиться, а то, без чего нельзя обойтись". Дизайн в концепции ХП следует этому принципу. Довольно часто, чтобы "улучшить" программу, разработчики добавляют массу возможностей, которые никогда не пригодятся пользователями или которыми будет пользоваться столь небольшое количество людей, что знай об этом разработчик - он не стал бы тратить время на их реализацию. Фразу "Это понадобится нам в будущем..." в ХП очень не любят - такие задачи могут сожрать до 90% рабочего времени всей команды, а польза от них не гарантирована. Очень большим талантом считается способность выбрать наиболее реальные задачи в проекте и сосредоточится именно на их решении.

КОДИНГ В ХП

Философия ХП затрагивает и этот аспект выполнения проекта. Даже если техническое задание правильно поставлено, разделено на части, распределенные между программистами, которые знают свое дело, проблемы все равно могут иметь место. Уволился или заболел сотрудник, который написал половину кода. Или одному программисту надо воспользоваться модулем, который написан другим. Или заказчик, увидев программу на промежуточной стадии ее разработки, вдруг воскликнул: "О боже, это ведь не то, что я хотел!". На этот случай, как ты уже догадался, в ХП тоже имеются свои правила.

Во-первых, "нельзя отпускать заказчика далеко". При составлении технического задания может случиться так, что заказчик что-то плохо объяснил в карточках пользователя или кто-то его не так понял. Поэтому частые релизы и консультации с заказчиком по ходу проекта могут сэкономить уйму времени - не придется переписывать то, что сделано "не так". Во-вторых, нельзя допускать, чтобы каким-то отдельным куском программы владел только один программист. Если он не сможет работать, то в его творении другой будет разбираться очень долго, что чревато неустойками и прочими проблемами с заказчиком. В идеале каждый кусок кода должен быть написан "парным программированием", то есть его должны писать два человека за одним компьютером. Один набирает, другой смотрит и советует. Периодически они меняются ролями. Такой подход часто оправдывает себя, потому что одна голова хорошо, а две лучше, к тому же парам проще найти удачное решение. К тому же у кода не будет слабых звеньев - кусочков кода, понятных только их автору.

РАБОТА В КОМАНДЕ

■ ХП - это прежде всего командный стиль программирования, созданный для команд и в котором командному духу уделяется особое внимание. В этом плане в ХП есть множество методов, которые нельзя не упомянуть.

Взаимовыручка.

Если к тебе за помощью обратился коллега - помоги. Бывает, что много драгоценного времени тратится впустую из-за того, что разработчик не видит какой-либо своей нелепой ошибки. Случайно поставленный или лишний символ может не заметить даже матерый программист, пусть даже проходил мимо них несколько часов. Свежий взгляд на проблему позволит сэкономить массу рабочего времени. К тому же так члены команды начинают лучше понимать то, чем заняты их коллеги.

Утренние собрания стоя.

Каждый день команды "экстремальных программистов" должен начинаться с такого собрания, на котором обсуждаются текущие планы. Стоя - чтобы не увлекаться и не засиживаться, а говорить по существу. Такие собрания позволяют любому сотруднику команды получить представление о том, что делает другой, и высказывать свои идеи и предложения для чужого сектора.

ЮНИТ-ТЕСТЫ

■ Еще одно нововведение ХП. Одновременно с разработкой программного модуля пишутся и модуль для его тестирования. На первый взгляд может показаться, что это лишняя работа, смысл которой выяснить невозможно. Однако в основе юнит-тестов лежат реальные потребности.

Иногда, меняя один из модулей программы, ты каким-то неявным образом затрагиваешь другой. И при этом можно потратить уйму времени на поиск "того, из-за чего все не работает". В ХП в этом случае запускаются юнит-тесты. Они выявляют те модули, которые работают с ошибками и облегчают процесс их поиска. К тому же когда в команду вливается новый человек, ему может быть полезно убедиться, что изменения, которые он внес, не изменили целостности всей программы: большую часть всего проекта он мог и не видеть. Юнит-тесты также очень сильно облегчают процесс рефракторинга.

РЕФАКТОРИНГ

■ Есть золотое правило программиста: если это работает - не трогай это. Экстремальные программисты на то и экстремальные - они его всячески нарушают. Рефракторинг - это о процессе, в котором код подвергается постоянной оптимизации и всяческим улучшениям. "Безжалостно рефракторить!" - вот один из девизов ХП. Вообще идея "А не переписать бы тут все..." не нова и часто приходит в голову одному разработчику, когда он смотрит на код другого. При этом процесс переписывания может продолжаться неоправданно долго и в конечном итоге не принести результатов - его бросят и оставят так, как есть, потому что "то, что было, хотя бы работает". Но как ты уже догадался, у грамотного рефракторинга тоже есть свои принципы.

Главный из них: код в конце каждого дня должен оставаться рабочим. Другими словами, нельзя углубляться и менять "сразу и все". Изменения стоит вносить небольшими порциями так, чтобы весь код в целом сохранял свою работоспособность. Написанные юнит-тесты облегчат для тебя выяснение того, не привело ли твое изменение в коде к каким-либо фатальным последствиям. Правил рефракторинга значительно больше, и про них можно написать отдельную статью. Если тебе интересно, я думаю, ты уже знаешь ключевое слово, которое надо набрать в Яндексе.

КОГДА ИСПОЛЬЗОВАТЬ ПРАВИЛА ХП

■ Я огласил далеко не весь "свод правил экстремального программирования". Их больше - подробнее можешь почитать на сайте или купить книгу. Однако крайне редко при разработке проекта используются все правила. Иногда из-за жестких сроков, иногда из-за того, что использование ХП требует больших трудозатрат и, как следствие, приносит меньше прибыли. Бывали случаи, когда написание юнит-тестов требовало больше времени, чем написание самой программы. К тому же при внесении изменений в программу приходится изменять и юнит-тесты...

Как видишь, ХП - не панацея от всех бед, и встречаются ситуации, когда использование всех ее правил потребует больше времени и сил, чем можно было сэкономит в дальнейшем. В общем, решать тебе. Советую помнить одну народную мудрость, которую любят в ХП: "Никогда нет времени, чтобы сделать сразу как следует, но всегда находится время переделать потом". Да, это философия больших проектов. Но некоторые ее правила ты можешь использовать и при работе в одиночку. Скажем, правила рефракторинга или написание юнит-теста для своей небольшой программы.

ХП - это не набор программ, а свод правил. Кроме правил, есть еще и программные средства, упрощающие и оптимизирующие работу программистов, потому что в конце концов доска с бумажными карточками в век информационных технологий выглядит как-то несолидно...

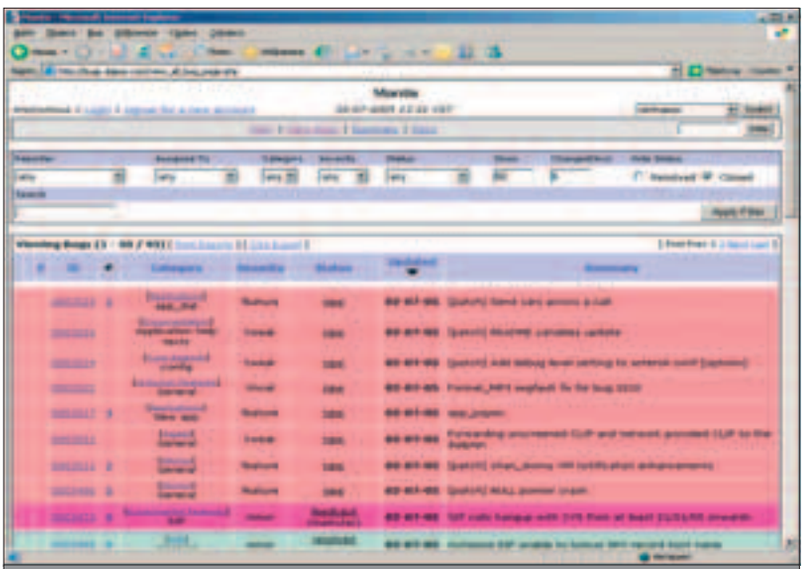
XPLANNER

■ Для автоматизации процесса программирования по стандартам ХП существует очень хорошее средство XPlanner, специально предназначенное для экстремальных программистов. Если ты работаешь в духе ХП, то это лучшее средство, в этой программе "реализованы" все принципы экстремального программирования. Она позволяет автоматизировать работу, даже если менеджеры, заказчики и программисты находятся на разных полюсах земного шара. Звучит заманчиво? Тогда тебе сюда: www.xplanner.org.

BUG TRACKING SYSTEMS

■ Как ты, наверное, догадываешься, программы живут по-разному. Некоторые пишутся ради "Дня Х", когда они должны отработать и после которого о них навсегда забудут. Некоторые программы будут использоваться годами и потребуют доработок и новых релизов. Для того чтобы автоматизи- »

XPlanner - мощное средство для экстремального программиста



Mantis – очень распространенная система Bug Tracking'а в интернете

ровать работу над такими длительными проектами, и были придуманы различные системы Bug Tracking'а, в которых пользователи программы могут сообщить о неисправности или о необходимости улучшений. Служащие технической поддержки могут выбрать, к какому разработчику или отделу разработчиков относится эта проблема. А непосредственно разработчик будет рапортовать о ходе выполнения этой задачи.

Некоторые такие системы чем-то напоминают форумы в интернете. Ты, как пользователь, найдя ошибку, заходишь, оставляешь сообщение с ее описанием. Его рассматривают, и если проблема действительно кроется в разработчиках, а не ты "не туда нажал", этой проблемой будут заниматься. Ответственный может обратиться к тебе за необходимыми уточнениями. Например, чтобы узнать, какую версию программы ты используешь или какая у тебя операционная система.

Эти программы сильно облегчают взаимодействие пользователей и разработчиков. С их помощью можно также контролировать разработчиков, следить за количеством багов, скоростью их устранения и т.д. и т.п.

Реализаций подобных систем множество. В виде форумов технической поддержки (например, wiki http://c2.com/cgi/wiki?WikiWikiWeb) или каких-то внутрикорпоративных программ (XPlanner или какая-нибудь собственная разработка компании), где задачи ставят работники этой же компании, являющиеся или пользователями, или сотрудниками службы технической поддержки. Эти же системы играют еще одну немаловажную роль: на основе их логов можно рассчитывать долевой вклад программистов в решение той или иной задачи.

СРЕДСТВА ПОСТАНОВКИ ЗАДАЧ

Постановка задач в любом проекте - это одно из необходимых условий успеха. Есть поговорка: если задача поставлена некорректно, то в результате может получиться вообще все что угодно. Безусловно, можно составить техническое задание на листике за пять минут. Возможно, вы с заказчиком настолько хорошо понимаете друг друга, что тебе и этот листик не нужен – все и так будет сделано в лучшем виде. Если все не так радостно, имей в виду, что серьезные задачи могут потребовать мощных средств постановки задач.

UML (UNIFIED MODELING LANGUAGE)

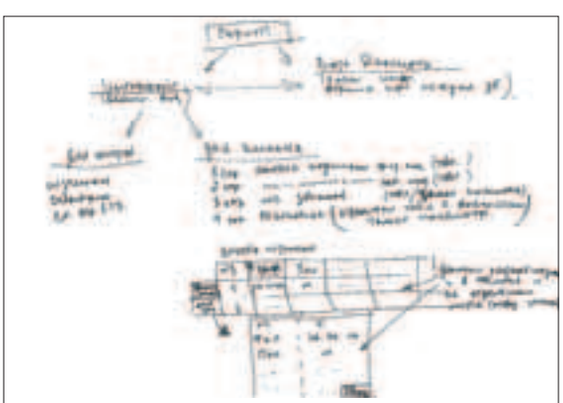
Это стандарты для проектирования интерфейсов и функционирования серьезных программ. Не буду углубляться в его описание – в интернете он и так слишком подробно описан. Посмотри сам, как выглядят прототипы программ в разных областях программирования, и сам сделай выводы о том, по каким техническим заданиям лучше работается.

Существуют и другие средства. Для некоторых областей программирования есть специальные средства и наборы стандартов. Большинство из них специально затачиваются под ту или иную область. Для проектирования баз данных, банковских процессов, Java-программ такие средства уже существуют практически для всех видов программирования. Если все же говорить о "простых" и "средних" средствах, то Word Excel и Visio – то, что доктор прописал.

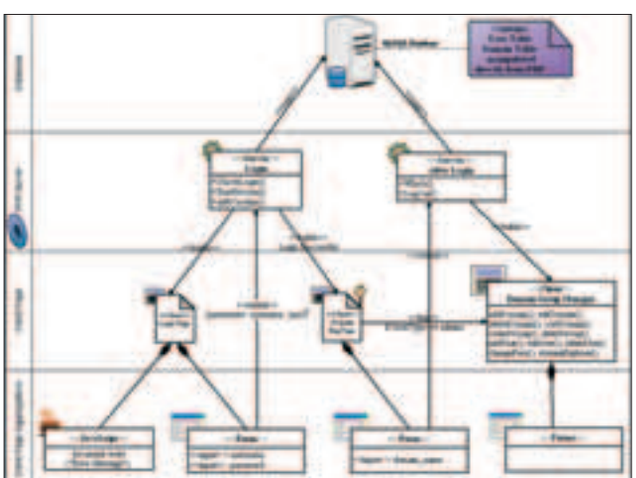
И еще один маленький совет. Уж лучше заставить человека написать подробное техническое задание, чем переписывать несколько раз "то, что поняли не так".

CVS (CONCURRENT VERSION SYSTEM)

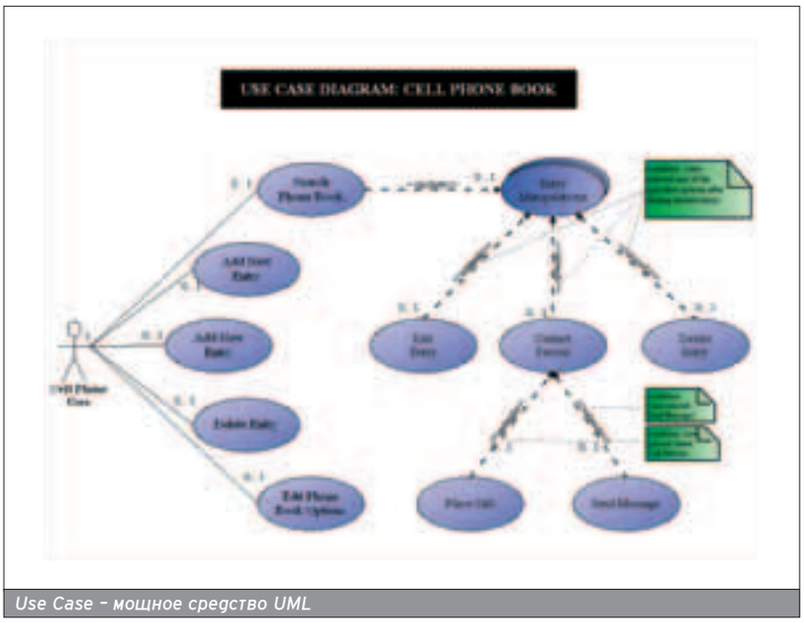
Система контроля версий, если по-русски. Чтобы ты понял, что это такое и для чего это предназначается, опишу несколько "классических ситуаций". Ты разрабатываешь какую-то программу. Выпустил релиз и продолжаешь работу над следующей версией, и вдруг в выпущенном релизе обнаруживается серьезный баг, а выпуска нового в ближайшем будущем не предвидится. Надо вернуться к выпущенной версии, исправить баг, затем вернуться к разработке новой. А в новой версии придется исправлять этот же баг уже во второй раз... Если над одними и теми же файлами работают несколько человек, конфликты между участниками этого коллективного труда неизбежны. Казалось бы, это можно решать криками "Вась, ты это файл трогаешь? Нет? Ну тогда и не трогай, я



Вот таких "технических заданий" стоит опасаться



UML в действии



Use Case – мощное средство UML

сейчас его править будут!". Звучит смешно, но, например, я говорил это тысячи раз, а слышал еще чаще. Каждый релиз можно складывать в отдельную папочку, исправления багов вносить сначала в выпущенную версию, а потом в разрабатываемую. А если разработчики не сидят в одном помещении, вообще разделены часовыми поясами и, соответственно, могут работать в разное время? При всем этом на "исправление бага" ты уже потратил много времени, неужели для его исправления в следующей версии придется начинать все сначала? Можно и дальше продолжать изобретения велосипеда. Однако такие проблемы решают уже давно, значит, должны существовать их решения.

Как ты уже понял из названия, CVS – это система, используемая для управления версиями файлов. В нашем случае - исподниками программ. Она используется для того чтобы хранить не только текущую версию файла, но и его "историю". В CVS хранится история всех манипуляций с файлом: когда он был добавлен в проект, когда и кем был изменен, а также какие именно изменения были внесены.

CVS – это клиент-серверное приложение. На сервере CVS хранятся все версии файлов, а клиент, как правило, имеет только одну – ту, с которой работает. После того как программист закончил вносить необходимые изменения в файл, он отправляет его CVS-серверу, который проверяет, нет ли ошибок (об этом читай ниже), и, если все в порядке, кладет файл на сервер и присваивает ему новую версию.

Возможно, над одним и тем же файлом работают несколько разработчиков, которые сами об этом не подозревают. Один открыл файл утром, работал с ним весь день и сохранил вечером. Другой внес небольшие изменения в середине дня. Если не воспользоваться системой контроля версий, то изменения, которые внес второй, пропадут, когда первый сохранит свои. При использовании CVS в конце дня, если первый разработчик соберется сохранить изменения, CVS-сервер выдаст предупреждение: "Вы работаете со старой версией файла. В CVS уже находится новая, пожалуйста, обновите ее из CVS".

CVS позволяет также отделаться легким испугом от другой ситуации, когда в выпущенном релизе программы обнаружена серьезная ошибка, а текущая версия программы еще не готова. Без использования CVS надо хранить все файлы "релиза" в отдельных папках и вносить изменения и в них, и в текущую версию. CVS значительно упрощает решение таких проблем, все файлы релиза можно пометить, и потом будет легко достать из CVS-сервера любой релиз. При этом если файлы релиза надо изменить не трогая их текущей версии, в CVS можно создать "ветвь". При этом текущая работа над новой версией будет продолжаться в обычном режиме и необходимости вносить изменения в него не будет. А когда будет готов очередной релиз, CVS поможет "слить ветви". Таким образом в новом релизе будут учтены изменения, которые были сделаны в предыдущем. Ну и еще одно неоспоримое преимущество CVS: так как последняя версия всех файлов находится на сервере CVS, работать над проектом могут люди из разных офисов, городов и даже стран.

Не обязательно прямо сейчас все бросать и, проникнувшись идеями CVS, приступать к изучению мануалов с cvs.ru. Для небольших проектов CVS может оказаться пустой тратой времени. Пока ты научишься пользоваться ей, настроишь сервер, тебе в голову может прийти такая мысль: "Если бы я не знал о CVS, я бы уже давно все написал". Это мощное и часто незаменимое средство. Если для проекта не планируется масштабных доработок и новых версий, то использование CVS – лишь трата времени. Кстати, CVS - не единственное средство контроля версий, есть и другие. Однако самое широкое распространение получила именно она, да и принципы у них схожи. Напоследок скажу, что CVS сейчас распространяется все шире и "умение работать с CVS" все чаще фигурирует в призывах устроиться на работу.

ТО, ЧТО НЕЛЬЗЯ АВТОМАТИЗИРОВАТЬ

Представь, что какой-нибудь злодей решил контролировать присутствие программистов на рабочих местах. Что тут можно сказать? Контролировать работу программистов с помощью программ...

В одной из компаний решили заставить сотрудников слать письмо по электронной почте при приходе на работу. О том, как заставить The Bat отправить письмо в определенное время, на следующий день знала даже секретарша. В другой известной мне компании время твоего прихода на работу фиксировалось по времени входа в корпоративную программу. "Приходитель вовремя, версия 1.0" был написан за два дня. Так что имей в виду: все автоматизировать не получится, кое-где останется неподкупная "тетя Маша", которая будет отмечать приход и уход каждого сотрудника.

ДУМАЙ!

Я описал разные технологии и подходы к программированию. Есть такая поговорка: если человеку дать в руки молоток, то для него все вокруг начнет подозрительно казаться гвоздями". Это я о том, что следует хорошо подумать перед применением той или иной технологии и выяснить, есть ли в ней реальная потребность.

Окупятся ли трудозатраты по написанию юнит-тестов? Стоит ли подвергать рефракторингу кусок кода, которому осталось работать несколько месяцев? Надо ли использовать CVS для задачи средних размеров, над которой ты работаешь на пару с товарищем? Если ты сейчас перечитаешь вопросы из введения и сравнишь их с этими, то, возможно, увидишь между ними одну существенную разницу: в вопросах введения спрашивалось, как решить задачу, а в вопросах заключения – стоит ли решать такую задачу этим методом. Если ты это видишь и понимаешь (на что я надеюсь), значит, ты что-то почерпнул из этой статьи. А еще говорят, что для того чтобы сделать правильный выбор, надо прежде всего иметь в распоряжении то, из чего будешь выбирать. Хорошо представляя себе возможности того или иного средства разработки, его преимущества и возможные проблемы, связанные с ним, ты сможешь решить, "стоит или не стоит" им пользоваться для решения этой задачи.

BUCKSWARE

Islander (islanderx@mail.ru)

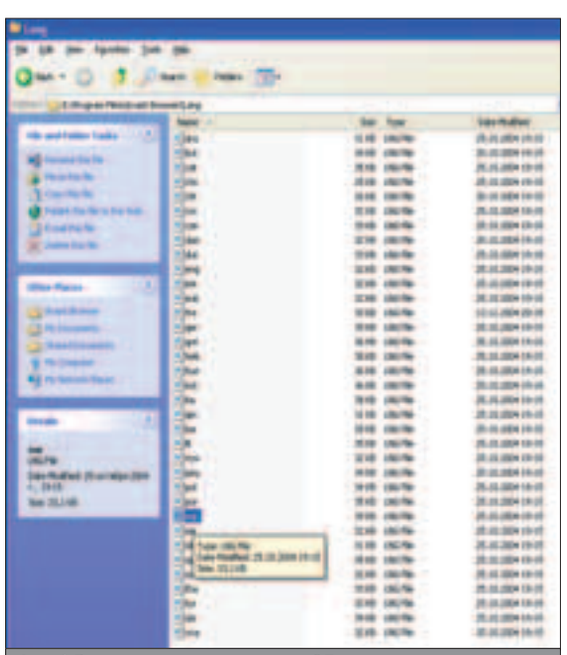
ПРОГРАММА-ПОЛИГЛОТ

УЧИМСЯ ЛОКАЛИЗАЦИИ ПРОГРАММ

Очень важной характеристикой программы является поддержка языков. Именно от этого зависит то, насколько распространится твой софт за рубежом, и, следовательно, величина дохода. Я постараюсь помочь тебе разобраться в технологиях и методах перевода и локализации ПО

Ты уже, наверное, начал писать свою программу, а по ночам тебе снятся горы денег, которые ты заработаешь на ней. Доллары... Доллары... Прочитав FAQ, кучу статей, обследовав со всех сторон своих конкурентов и определив бюджет, ты понял, что русский рынок - это не для тебя. На нем можно попытаться заработать денег, и некоторые делают это успешно, но твоя программа достойна того, чтобы о ней узнал весь мир.

Тогда представь себе привередливого буржуя с толстым кошельком, который вместо того чтобы учить русский язык и пользоваться твоим продуктом, хочет непременно получить его английскую версию. Да, что поделаешь, горькая правда... Программу надо переводить. Достаточно ли перевода только на английский язык? Для начала да. Если твоя программа станет популярной, то пройдет буквально две-три недели и ты получишь трогательное письмо от забугорного, но без гроша в кармане студента, который попросит у тебя ключик к программе в обмен на перевод интерфейса. Советую сразу соглашаться, так как перевод интерфейса - обычно дело весьма долгое, нудное и дорогостоящее.



Avant Browser поддерживает аж 35 языков

Надеюсь, теперь ты уже не сомневаешься, что тебе нужен многоязычный интерфейс, и уже представил свою программу говорящей на 24-х языках... Мечты, конечно, но это не так далеко от реальности - сделаем первый шаг!

Среднестатистический shareware-продукт обычно представляет собой небольшую программу, написанную на MS Visual C, Borland C++ Builder или Borland Delphi. Наши шароварщики особенно любят последнюю, поэтому я буду рассматривать перевод в контексте Delphi, однако для других сред и языков все это будет выглядеть примерно так же, и ты легко адаптируешь под них полученные знания.

Есть несколько способов хранения перевода. Первый заключается в том, что строки "жестко" зашиваются в код программы. Однако у этого способа есть один большой недостаток: его очень сложно модифицировать. То же самое можно сказать и про разные встроенные средства перевода - лучше их не использовать. Остановимся на двух способах, горячо любимых нашими шароварщиками: это вынос перевода в lng-файлы и "зашивание" перевода в ресурсы внешних dll-библиотек.

Какой из этих способов выбрать? Если ты планируешь, что версии программы на разных языках будут отличаться исключительно строками, смело выбирай lng. Если же ты хочешь, чтобы у них были еще и разные ресурсы (графика, звук и проч.), то придется остановиться на варианте с ресурсами в dll. Вот уж у бедного студента-чужестранца прибавится работы! Можно облегчить его участь и научить его пользоваться Restorator'ом. В общем, если кажется слишком мутно, посмотри на таблицу.

ПЕРЕВОД С ПОМОЩЬЮ LNG(INI)-ФАЙЛОВ

Перевод с помощью ini-файлов (обычно разработчики меняют расширение файла на lng, однако его структура идентична ini-файлу) является самым популярным в классе небольших программных продуктов. Ini-файл представляет собой обычный текстовый документ, разбитый на секции. В каждой секции есть параметры и значения этих параметров. Пример ini-файла легко обнаружить в каталоге Windows.

Названия секций в файле заключаются в квадратные скобки, регистр символов не имеет значения. В каждой секции присутствуют параметры и их значения, которые разделены знаком "равно" по формату param=value. Полезная возможность ini-файлов - поддержка комментариев. Все комментарии начинаются со знака ";", после которого ты можешь писать все, что заблагорассудится.

Посмотрим, как все это будет выглядеть на практике. Строки обычно логически относятся к той или иной форме приложения, поэтому было бы разумно сгруппировать их по этому признаку.

Я рекомендую использовать префиксы для всех элементов управления на форме (например, для кнопки btn, для метки lbl), а для сообщений, которые будут показаны пользовате-

Возможность	Перевод в lng-файлах	Перевод в ресурсах dll	Перевод, «зашитый» в программу
Хранение строковых значений	Да	Да	Да
Хранение любых видов ресурсов, будь то графика, звук или анимация	Нет	Да	Да
Удобство редактирования	Высокое	Низкое	Очень низкое
Возможность менять данные без перекомпиляции приложения	Да	Да	Нет

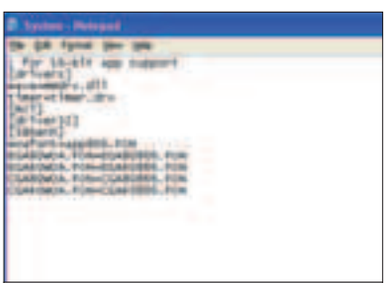
Сравнение методов хранения перевода

лю с помощью MessageBox'ов, использовать параметры с префиксом msg. Для всех прочих строковых меток будет использоваться префикс str. Что же делать со строками, которые логически не относятся ни к одной форме? Например, сообщение об ошибке проверки контрольной суммы exe-файла на том этапе, когда ни одна форма еще не создана. Все очень просто: создадим еще одну секцию, назвав ее, например, Global, и будем сваливать туда эти строки. Можно поступить подобным образом, если какие-нибудь ресурсы с префиксами msg и str используются одновременно в нескольких формах: запихнем их в секцию COMMON.

Также не стоит забывать, что в начале ini-файла желательно указать версию приложения, для которой разработан этот файл, а также название языка и имя автора перевода.

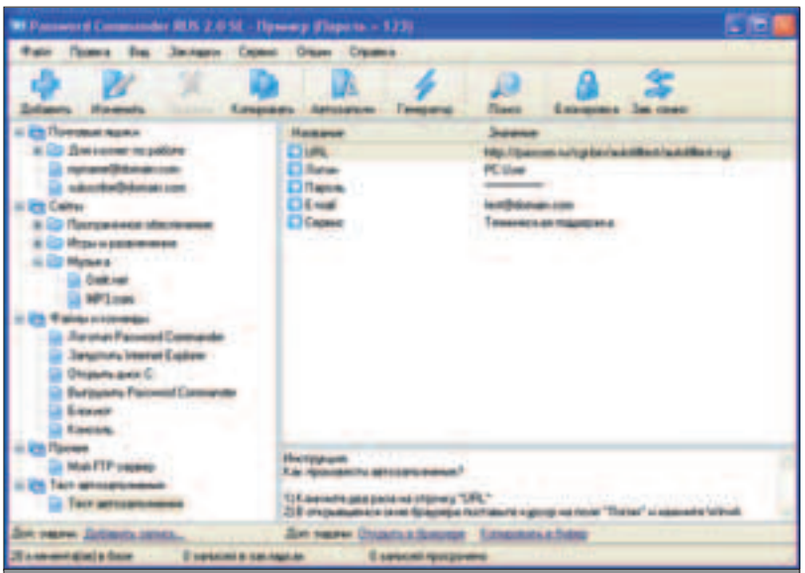
Некоторые разработчики, например, Кристиан Гиспер (автор небезызвестного Total Commander'а) дают параметрам менее говорящие названия, например:

```
0="Нет доступа, или файл\n%s\n уже используется."
1="Укажите шаблон(ы) файлов (например:
s*.doc;*.tx?)"
2="Создать новый каталог:"
```



Так выглядит файл system.ini, если открыть его в "Блокноте"

Так выглядит lng-файл серьезной многоязычной программы



Популярный менеджер паролей Password Commander хранит переводы как раз в lng-файлах

На мой взгляд, в наше время такой подход неоправдан. Лучше пусть lng-файл будет вместо 30 Кб занимать 45, зато и тебе, и переводчику будет сразу понятно, что к чему относится.

Формат хранения мы разобрали, теперь рассмотрим технологию и алгоритм перевода. Мы не будем уподобляться программам, которые при смене языка интерфейса требуют перезапуска, а сделаем переключение языка "на лету". Для этого нам потребуются две процедуры: одна для перевода всех компонентов формы, а другая для перевода любой строки по названию ее параметра.

```
Procedure TranslateFormComponents (Form:
TCustomForm; LngFileName: string);
var
  I : integer;
  IniFile : TMemIniFile;
  S : string;
begin
  IniFile := TMemIniFile.Create (LngFileName);

  for i := 0 to Form.ComponentCount - 1 do
  begin
    if Form.Components [i].name = '' then
    Continue;
    S := IniFile.ReadString (Form.Name,
Form.Components [i].name, '#_ERROR_#');

    If S <> '#_ERROR_#' then
    begin
      If Form.Components [i] is TLabel then TLabel
(Form.Components [i]).Caption :=S;
      If Form.Components [i] is TButton then TButton
(Form.Components [i]).Caption :=S;
    end;
  end;
  IniFile.Free;
end;

Function TranslateMessage (SectionName, MsgID,
LngFileName: string):string;
var
  IniFile : TIniFile;
begin
  IniFile := TIniFile.Create (LngFileName);
  Result := IniFile.ReadString (SectionName, MsgID,
'#_ERROR_#');
  IniFile.Free;
end;
```

Мы имеем две функции, и это практически все, что нужно для перевода. Создаем в классе каждой формы приложения процедуру ApplyTranslate, в которой вызываем TranslateFormComponents (self, 'Имя_нашего_lng-файла'), а также переводим все прочие статические элементы управления, которые не могут быть переведены автоматически. Не стоит также забывать и о всплывающих сообщениях(MessageBox'ах и т.п.), которые переводятся очень просто. Например, если тебе надо показать сообщение о том, что выйти из программы в текущий момент невозможно, пишешь вот так:

»

```
ShowMessage (TranslateMessage (self.name,
'msgCannotExit', 'Имя_нашего_lng-файла'));
```

Если такая запись покажется слишком длинной и громоздкой (а мне так и показалось), можно сделать проще. Создай в классе формы новую функцию TranslateMsg и используй ее:

```
Function TranslateMsg (MsgID: string):string;
begin
  Result := TranslateMessage (self.name, msgID, 'Имя_нашего_lng-файла')
end;
```

Вывод сообщения при этом примет такой вид:

```
ShowMessage (TranslateMsg ('msgCannotExit'));
```

Красота! Теперь при переключении языка интерфейса не забывай вызывать ApplyTranslate у формы и добавлять в lng-файл новые строчки по мере разработки проекта.

Небезызвестная программа "Соло на клавиатуре" хранит перевод в ресурсах dll-файлов

ПЕРЕВОД С ПОМОЩЬЮ "ЗАШИТЫХ" В DLL РЕСУРСОВ

Допустим, у тебя есть картинки, звуки и прочие мультимедийные приблуды, которые в ini-файл никак не засунуть. Что ж, придется запихивать их в ресурсы внешних dll-библиотек. Редактировать файлы ресурсов не так удобно, в "Блокноте" их особо не попишешь, поэтому для такой цели одни используют встроенный редактор ресурсов в Visual Studio, другие – старый добрый Restorator, а ты, может быть, найдешь еще что-нибудь более удобное.

Логика работы с ресурсами в dll почти такая же, как и при переводе с помощью ini-файлов, но здесь присутствуют некоторые характерные особенности. Например, если хранение строк сообщений и динамических надписей (msg и str) ни у кого не вызывает вопросов, то с хранением форм будут проблемы. Можно в design-time создать форму, перевести на ней все надписи и запихнуть ее в ресурсы dll, а потом читать оттуда. А можно оставить все формы в ресурсах приложения, а в dll поместить только названия элементов управления, что и было сделано при работе с lng-файлами.

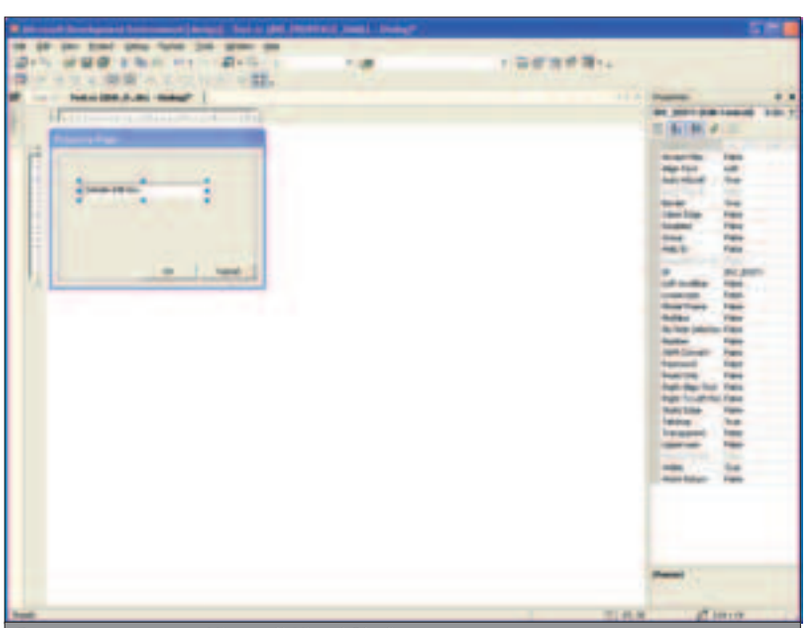
Подключать динамические библиотеки с переводом нужно, естественно, динамически. Делается это элементарно с помощью функции LoadLibrary.

Рассмотрим разницу между реализацией перевода через ini-файлы и ресурсы в dll.

Если ты собираешься помещать в ресурсы целую форму, то функция TranslateFormComponents тебе, естественно, не пригодится. В противном случае нужно выковыривать строки из ресурсов dll. Это не самое веселое занятие, так как ты не сможешь разбить строковые ресурсы на секции, а каждый строковой ресурс в dll должен быть уникален. Итак, в TranslateFormComponents ты должен сначала загрузить dll-библиотеку с помощью LoadLibrary, а затем получить нужный строковой ресурс функцией LoadString. В конце не забудь вызвать FreeLibrary.

Я советую использовать префиксы для всех ресурсов dll по формату: название формы "_" имя ресурса. Например, если у нас на форме с именем frmMain имеется кнопка btnExit, то название строкового ресурса будет выглядеть так: frmMain_btnExit. С помощью такой системы префиксов названия ресурсов будут удобно отсортированы, и добавление нового не составит большого труда. Имя ресурса с такими префиксами в функции TranslateFormComponents будет выглядеть следующим образом:

```
Form.name + '_' + Form.Components [i].name
```



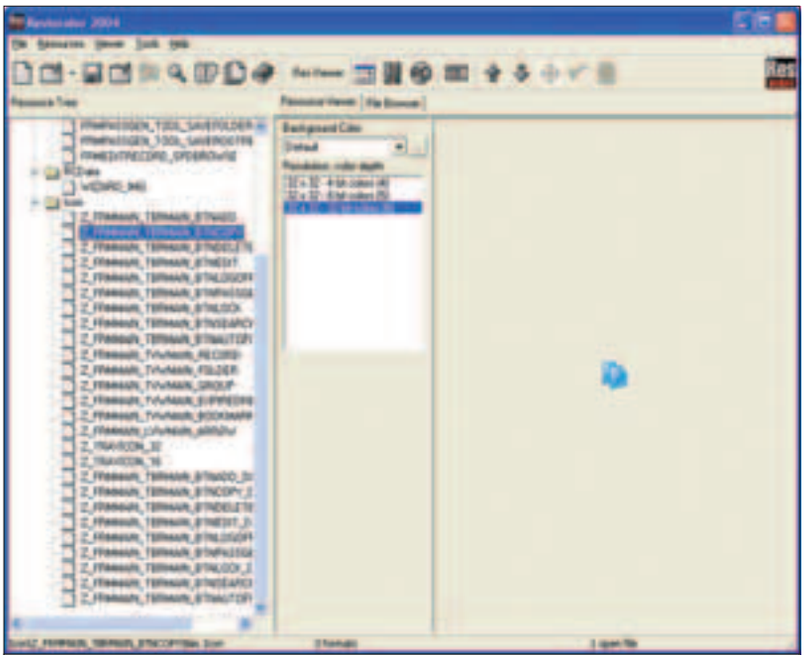
Редактор ресурсов в Microsoft Visual Studio

Еще одна тонкость - обязательно проверять наличие ресурса перед тем как загрузить его, используя функцию:

```
HRSRC FindResource(
        HMODULE hModule,
        LPCTSTR lpName,
        LPCTSTR lpType);
```

Иначе будут вылезать exception'ы, если вдруг бедный студент-чужеземец схалтурит и не переведет пару строчек.

Аналогично функции TranslateFormComponents перепиши TranslateMessage. Далее все абсолютно так же, как и с lng-файлами: создаешь процедуру ApplyTranslate в классе формы и вызываешь ее по мере необходимости. Для загрузки графики ты можешь использовать функции LoadIcon и LoadBitmap, подробное описание которых можно найти в



Restorator - один из лучших редакторов ресурсов

MSDN. Я думаю, ты все понял и дальше разберешься сам. Только не забудь обучить бедного забугорного студента пользоваться Restorator'ом!

Юникод

Если твоя программа должна поддерживать возможность работы более чем с двумя различными языками сразу или же обеспечивать поддержку экзотических языков, стоит подумать о том, чтобы сразу оснастить свой продукт поддержкой юникода. Тут не все просто: если в Windows 2k/XP/2003 юникод полностью поддерживается на уровне ядра, то в Windows 9x/ME возникнут трудности. Поэтому придется определиться с тем, должна ли твоя программа работать под Win9x. Однозначных советов по этому поводу давать не стану, но скажу, что поддержка Win9x все еще очень желательна для утилиток массового использования и сравнительно недорогих.

Если же твоя программа будет работать только в Windows 2k/XP/2003, то в MS Visual C++ тебе будет достаточно определить макрос UNICODE в настройках проекта и все элементы управления автоматически начнут поддерживать юникодные надписи. В Delphi/Builder будет немного сложнее: без дополнительных библиотек тут не обойтись. Рекомендую скачать TNT Unicode Controls и использовать элементы управления только из этой библиотеки, так как они полностью поддерживают юникод. Библиотека бесплатная и распространяется с открытым исходным кодом. Можно скачать ее тут: http://download.tntware.com/delphi_unicode_controls/TntUnicodeControls.zip.

Если твоя программа должна поддерживать юникод в Windows 9x/ME, то тут все гораздо сложнее. В MS Visual C++, кроме макроса UNICODE, надо будет использовать специальную библиотеку unicows.lib, однако всех проблем она не решит. Например, все меню и заголовки окон придется перерисовывать вручную. Эта тема очень обширная, поэтому советую почитать в MSDN раздел Microsoft Layer for Unicode. Delphi/Builder создаст проблем. TNT Unicode Controls под Win9x/ME уже не выручат, так что надо искать что-нибудь другое. Сразу хочу тебя огорчить: мне не удалось найти бесплатного пакета юникодных компонентов на все случаи жизни. Может быть, ты сможешь? Я остановил свой выбор на пакете ElPack (www.lmd.de/products/lmdelpack). Стандартная версия стоит $99, ее вполне хватит для среднего шароварщика.

Интерфейс

Очень часто возникают следующие проблемы: в русском переводе все нормально, а, допустим, в немецком строки получаются слишком длинные и залезают на другие элементы управления, вылезают за края формы и т.д. Что делать? Тут я дам два совета: во-первых, почти любую фразу можно сократить, во-вторых, учись проектировать интерфейсы. К сожалению, наши программисты известны всему миру навороченными донельзя программами и полным отсутствием usability. Поэтому, если ты не умеешь проектировать интерфейсы, советую курить MSDN до просветления. Можешь начать курить отсюда: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/anch_UIDesignDev.asp. Если курить не хочется, можно заказать проектирование интерфейсов профессионалам. У нас есть несколько контор (например, www.imho.com.ua), которые этим занимаются и за определенное вознаграждение грамотно расположат элементы управления на форме.

Скрыпников Сергей (http://slammy.ru) и Павел Сурменок

.NET КОНКУРЕНТАМ!

ТЕХНОЛОГИЯ .NET НА ПАЛЬЦАХ

Про технологию .NET написано довольно много книг. В первой части этой статьи мы расскажем о плюсах и минусах "точки нет", а в во второй - вместе напишем простенький сканер портов, который может работать сразу в несколько потоков.

PART ONE. ТЕОРИЯ

ПОЧЕМУ .NET?

Технология .NET требует обязательного рассмотрения на страницах журнала, например, потому что компания Microsoft когда-то тратила около 70-80% (по различным источникам) своего бюджета на разработку этой технологии, после выхода технология получила широкий общественный резонанс в среде программистов, и положительных отзывов было неоспоримо больше, чем негативных. Третьей основной причиной можно назвать то, что dotNET широко используется для программирования web-сервисов/web-служб, а в наше время трудно найти человека, у которого бы не было потребности в работе с интернетом, да и за разработку сетевых программ платят больше (если, например, сравнивать одинаковый объем работы программиста сетевых служб и "обыкновенного" :). Стоит учитывать, что и до появления технологии .NET разрабатывались сетевые службы, но в одиночку этого, как правило, никто не делал (для каждого правила есть, конечно, исключения), а сейчас это сделать проще простого.

LET'S GO

Итак, перейдем непосредственно к самой .NET: чем она хороша и что в ней не устраивает нас. Будем придерживаться принципа "сложное - понятным языком", а в обильных технических терминах ты сможешь разобраться при дальнейшем изучении этой технологии. Начнем непосредственно с плюсов (здесь и далее будет в основном рассматриваться язык VB.NET, иногда в сравнении с VB 6.0).

ПЛЮСЫ

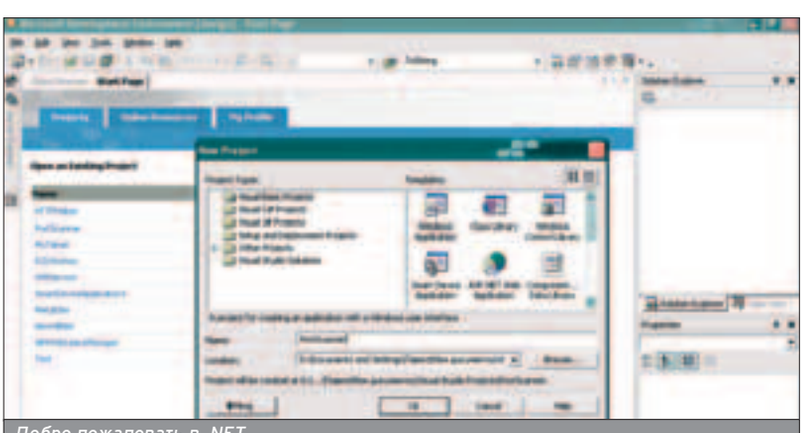
❶. "Мультиязычность" dotNET. Приложения можно писать на любом из нескольких десятков CLS-совместимых языков. Неполный список языков программирования ты можешь найти на одной из врезок.

❷. Упрощение разработки. Среда разработки Visual Studio .NET предоставляет очень удобные и мощные средства разработки приложений. Очень мощный IntelliSense (что-то наподобие помощника в написании кода). Например, программист написал: "Dim stream As New FileStream", а он ему покажет в всплывающей подсказке (hint - это по-нашему :)) перегруженные варианты синтаксиса.

Еще пример: написал - stream, а IntelliSense тебе вывесит в список "Все свойства/методы объекта". В общем, очень удобная штука, реально помогающая при написании программ (хотя некоторые ругаются матом :) и отключают эту фишку). Также присутствуют различного рода "мастера", которые точно так же облегчают эту нелегкую жизнь и тебе, и мне, особенно в работе с данными: фактически простенькую форму данных или решетку (grid) с привязкой к БД можно настроить только с помощью мастеров, причем не написав ни строчки кода. А если еще и пару строчек кода добавить, то получится совсем шедевр - будешь ходить и за деньги всем предлагать :).

❸. Цены и свободы. Если сравнить их, например, с Visual Studio 98, которая была монолитным и недешевым пакетом, то с .NET дело обстоит несколько свободнее. .NET Framework SDK, включающий основную документацию по .NET Framework, компиляторы языков VB .NET, VC# .NET, VC++ .NET, J# и др., а также ряд небольших утилит (например, визуальный дизайнер форм), поставляется полностью безвозмездно (можно свободно скачать с сайта http://microsoft.com). Таким образом, можно, не потратив ни копейки, писать программы, например, в "Блокноте" (хотя ОС в этом случае все равно придется покупать :)). Кроме того, существует ряд бесплатных средств разработки от сторонних производителей. Наиболее заметные из них: SharpDevelop (поставляется с исходным кодом) и WebMatrix - творение команды разработчиков ASP .NET, сделанное "на коленке" для каких-то своих нужд, но позже доработанное до приличного продукта; весит это чудо чуть меньше 2 Мб и даже включает в себя web-сервер для тестирования ASP .NET приложений. Сама VS .NET поставляется в широком спектре модификаций, имеется даже бесплатная, но немного урезанная версия - Express.

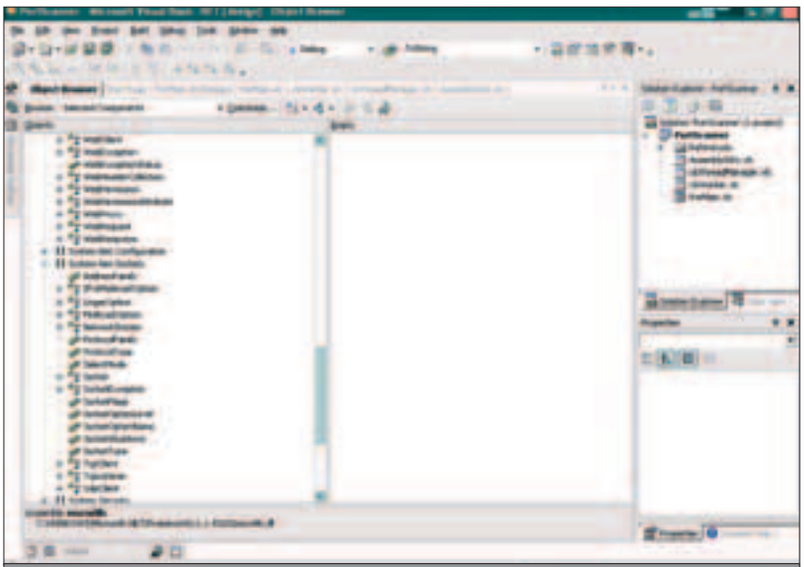
❹. Мощная библиотека классов .Net Framework Class Library содержит самые разные средства, которые могут понадобиться при разработке программ, таких как доступ к БД, создание интерфейса, работа с графикой (GDI+), XML (в том числе XPath, XSLT, XQuery), файловая система, криптография (шифрование, хэширование, цифровые подписи), сеть и многое-многое другое, о чем ты даже и не подозревал :).



Добро пожаловать в .NET

"МУЛЬТИЯЗЫЧНОСТЬ" .NET

Microsoft VB .NET
Microsoft VC++ .NET
Microsoft C#
Microsoft J#
Microsoft Jscript
APL
ASNA Visual RPGRPG.NET
Fujitsu COBOL
Micro Focus Cobol NetExpress
F# (a mixed functional/imperative anguage based on Caml from Microsoft Research)
Eiffel
Delta Forth
Lahey/Fujitsu Fortran for .NET
Salford Fortran
Hugs98 Haskell
Glasgow Haskell
ActiveState Perl.NET
ActiveState Python.NET
Mark Hammond Python.NET
Mercury.NET
Mondrian
Component Pascal
TMT Pascal
Tachy - subset of Scheme
HotDog Scheme
Smalltalk SmallScript
SML.NET
QKS Smalltalk [dead]
A#
Active Oberon for .net
AsmL
CAML
Delta Forth
Dyalog APL
Fortran for .NET
FTN95 for .NET
Haskell
ML
NetCOBOL for .NET
Perl
Python
SmallScript (S#)



Классы для работы с сокетами

❺. Потенциальная кроссплатформенность. DotNET-приложения первоначально компилируются в так называемый промежуточный код (MSIL - Microsoft Intermediate Language), который никак не зависит от "платформы", то есть от процессора и операционной системы. А компиляция в машинный код, зависимый от процессора (JIT-компиляция, Just-in-time-компиляция), происходит уже на машине конечного пользователя (другими словами, на машине дядьки, который отвалил тебе много кэш за твою очень сложную и долгую работу :)) при первом запуске программы. Это позволяет (в теории) создавать кроссплатформенные приложения, то есть заставить один и тот же бинарник запуститься и под PC, и под Macintosh. Сейчас энтузиастами ведутся разработки вариантов .NET FW под Linux, FreeBSD и Macintosh. Например, можно сказать о Mono - dotNET-среде для Linux, которая пока еще находится на стадии глубокой альфа-версии, но ходят слухи, что программировать под это чудо научной мысли уже вполне возможно. Конечно, настоящую живую кроссплатформенность мы пока не увидим, но "возможно все".

❻. ASP .NET. ASP .NET (web-приложения .NET) - отдельная тема для разговоров. Сотрудники компании Microsoft потрудились и создали отличную штуку, как бы смешно на первый взгляд это ни казалось! Теперь разработка web-приложения сравнима с программированием под Windows. Визуальный дизайн страниц, система событий и т.д.

❼. Обеспечение взаимодействия приложений. DotNET предоставляет на данный момент две технологии взаимодействия приложений: .NET XML Web Services и .NET Remoting. Web Services уже сейчас очень активно используются на крупных предприятиях. Например, интернет-магазин Ozon предоставляет web-сервис для получения информации о доступных в магазине книгах, что позволяет разработчикам других сайтов наладить получение этой информации и использование ее в своих корыстных целях (к примеру, на http://vbnet.ru в разделе "Магазин" список книг обновляется через этот web-сервис).

❽. Если ты счастливый обладатель Windows 2003, то для тебя первый минус - никакой и не минус :).

❾. Плюсы, показывающие себя во всей красе при создании кода :). О некоторых из них ты узнаешь в продолжении статьи, полностью все описать невозможно, так как изменений, повторяю, так много, что об этом уже пишу книги.

МИНУСЫ

Перейдем к минусам, потому что каждому известно, что в каждой бочке есть ложка, а в каких пропорциях, мы предлагаем тебе решать самостоятельно - здесь нужно опираться только на личные потребности и ощущения.

❶. Основным и достаточно весомым (во всех смыслах) аргументом против .NET является размер пакета .NET Framework, который необходим для работы .NET-приложений, то есть сама программа, например, может весить 100 Кб, но она не запустится, пока не скачать вышеназванный пакет (правда, скачать его нужно всего »

COMMON LANGUAGE SPECIFICATION (CLS)

■ Для улучшения взаимодействия между языками в Microsoft .NET Framework введен языковый стандарт, Common Language Specification (CLS). CLS - это поднабор свойств языка, поддерживаемых CLR, включающий свойства, общие для большинства объектно-ориентированных языков программирования. Если ты хочешь, чтобы твои компоненты и элементы управления можно было использовать из других языков программирования, нужно создавать их на CLR-совместимом языке и обеспечить совместимость всех общих и частных членов с CLR.
Источник: **www.rsdn.ru**.

один раз). Его последний релиз весит 24 Мб, что заметно затрудняет распространение приложений через сеть интернет (особенно в странах exUSSR, где со связью не все в порядке).

❼. С применением .NET невозможно создать такие же быстрые и компактные приложения, какие делают, например, в C++ или Assembler, что и понятно. Выше уровень программирования (а ассемблер, например - низкоуровневый язык программирования) - ниже скорость работы программ, но при этом заметно выше скорость их разработки. А что важнее для тебя, решай сам: если потенциальный заказчик разработки ПО требует, чтобы ему разработали приложение за рекордно короткое время, то, конечно, обращайся к .NET (хотя это не означает, что все остальное можно выкинуть в помойку). Если же он требует, чтобы программа запускалась и на "трешке" (трешка- это не PentiumIII (пояснение для очень молодых), то надевай очки с толстыми линзами и за ассемблер.

❽. Заявленная кроссплатформенность на момент написания статьи находилась еще на стадии разработки, и полноценно использовать все плюсы новой технологии пока возможно только для систем на ОС Windows (кстати, если кроссплатформенность все же удастся реализовать, то, возможно, позднее появятся кроссплатформенные вирусы, а счастливы будут те, кто никогда не скачивал себе .NETFW).

❾. Остальные минусы или совсем несущественны, или основаны не на объективной оценке.

КАЧАТЬ ИЛИ ПЛАТИТЬ?

■ Существуют частичные решения проблемы скачивания 24 Мб дистрибутива .NET FW для пользователей. Например, некая фирма RemoteSoft предлагает за абсолютно безумные деньги программу для прилинковки к exe'шнику только тех частей .Net Framework, без которых эта программа не сможет жить. В итоге размер дистрибутива уменьшается эдак до 3-5 Мб, но в фирме люди продуманные, поэтому продают программу по принципу "утром деньги - днем стулья" и даже Trial не предлагают: знают, что русские обязательно поломают :).

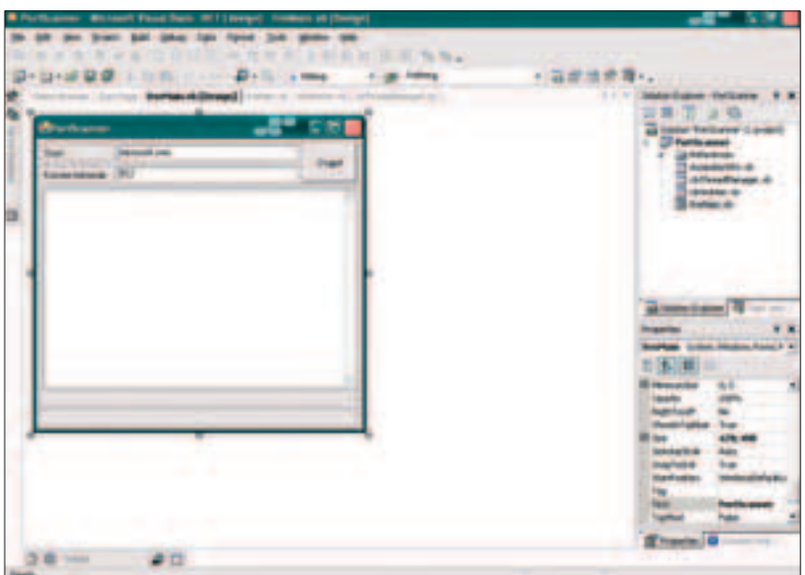
PART TWO. ПРАКТИКА

ПРОГРАММИРУЕМ НА VB.NET

■ Итак, с теорией ты уже разобрался и, наверное, хочешь посмотреть, как все происходит на практике. Мы долго думали, что же все-таки напрограммировать, чтобы показать некоторые средства языка/технологии, при этом уместиться на отведенную под статью площадь и понравиться тебе. Решили написать простенький сканер портов, который может работать сразу в несколько потоков. Начнем?!

СКАНЕР ПОРТОВ

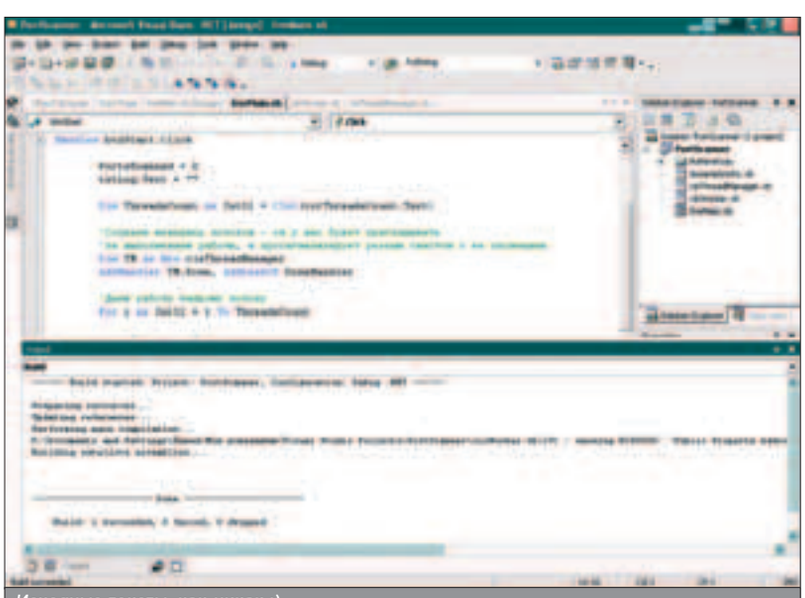
■ Открываем Visual Studio .NET (при разработке программы использовалась версия 2003 Enterprise Architect - последний релиз), создаем новое Windows-приложение. Начнем с интерфейса. Форму мы по привычке переименовываем в frmMain, пишем заголовок формы (в свойстве Text). Размещаем на форме контролы: два Label с надписями "Хост" и "Кол-во потоков", рядом с ними два TextBox с именами txtHost и txtThreadsCount, кнопка btnStart (надпись "Старт"). Ни-



Внешний вид нового Windows-приложения

же - большой TextBox txtLog (Multiline=True, ScrollBars=Vertical), внизу - два Label (lblProgress1 и lblProgress2). Располагать их, конечно, ты можешь по своему усмотрению. Выставляем для каждого элемента свойство Anchor, которое было задумано разработчиками .NET для авторесайза контролов при изменении размеров формы. У контролов с надписями "Хост" и "Кол-во потоков" Anchor=Left Or Top, у txtHost и txtThreadsCount Anchor = Left Or Top Or Right, у кнопки - Right Or Top, у txt Log -Left Or Right Or Top Or Bottom, и, наконец, у двух Label'ов внизу формы - Bottom Or Left Or Right.

Интерфейс готов. Теперь напишем класс clsWorker, который будет заниматься основной работой - сканировать указанные порты удаленного хоста (именно это было твоим самым



Исходные тексты, как-никак :)

сокровенным желанием, правда?). Для работы с сетью в .NET Framework задействован ряд классов в пространстве имен System.Net, а классы работы с сокетами лежат в System.Net.Sockets. Чтобы при каждом упоминании нужных классов не писать эти длинные приставки, можно импортировать пространства имен. Для этого в файле .vb вне декларации класса (в самом начале файла) нужно написать пару строк:

```
Imports System.Net
Imports System.Net.Sockets
IP хоста получаем с помощью метода GetHostByName класса DNS.
Dim IP As Int64 =
Dns.GetHostByName(Host).AddressList(0).Address
```

Далее для каждого порта, который нужно сканировать, создаем Socket и пытаемся соединиться. Если попытка соединения не удастся, будет сгенерировано исключение (Exception), которое мы проигнорируем вписав вызов Connect в конструкцию Try...Catch с пустым блоком Catch.

```
Dim socket As New Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp)
Try
socket.Connect(New IPEndPoint(IP, i))
Catch
End Try
```

После этого узнаем, удалось ли socket'у соединиться (свойство - IsConnected), и передадим эту информацию "куда следует" :), после чего закроем socket:

```
RaiseEvent Log(i, socket.Connected)
socket.Close()
```

ClsWorker взаимодействует со своим хозяином (классом frmMain) с помощью механизма событий. В clsWorker описано событие Public Event Log(ByVal Port As Int32, ByVal Opened As Boolean), которое вызывается из кода clsWorker в процессе работы. Так как сканер у нас многопоточный (а как же - фирма веников не вяжет ;), в программе используется класс System.Threading.Thread. Создаем экземпляр класса Thread, в конструкторе которого указываем делегат на процедуру, которая будет запущена в этом потоке (делегат - грубо говоря, ссылка на процедуру, которую можно передавать куда угодно, если пожелаешь, даже средствами .Net Remoting в другой процесс или на другой компьютер и через которую эту процедуру можно легко и просто вызвать). Теперь настраиваем нужные свойства объекта и, наконец, для запуска потока вызываем метод Start. Чтобы прервать выполнение потока, нужно вызвать метод Abort. Приостановить поток - метод Suspend. А для ожидания завершения потока используется метод Join (тот поток, который вызвал этот метод, останется приостановленным до тех пор, пока Thread не станет завершенным).

Теперь пришло время написать код формы frmMain. В первую очередь нужно объявить несколько переменных уровня класса: массив PortsList, в который будут записываться состояния портов, и переменная PortsScanned - счетчик проверенных портов.

```
Private PortsList(65535) As Boolean
Private PortsScanned As Int32
```

В обработчике события клика кнопки (btnStart.Click) пишем код запуска проверки портов. Заметим, что мы писали программу с учетом того, что в поле "Кол-во потоков" будут вводиться степени числа 2 (1, 2, 4, 8 и т.д.). Переделать код с учетом "некруглого" числа потоков мы тебе предлагаем самостоятельно, а заодно и проверку на корректность заполнения поля.

Итак, получаем введенное пользователем число потоков: Dim ThreadsCount As Int32 = CInt(txtThreadsCount.Text). Для учета потоков сделан класс ThreadsManager, который ждет завершения "рабочих" потоков и сигнализирует пользователю о завершении работы. Его мы рассмотрим далее. Создаем менеджера потоков и подписываемся на его событие Done (оно сработает при окончании работы).

```
Dim TM As New clsThreadManager
AddHandler TM.Done, AddressOf DoneHandler
Далее для каждого потока создаем рабочего, устанавливаем ему "трудовую норму" :).
Dim Worker As New clsWorker
Worker.Host = txtHost.Text
Worker.FirstPort = (i - 1) * 65536 / ThreadsCount
Worker.LastPort = i * 65536 / ThreadsCount - 1
```

Подписываемся на событие Log, которое будет информировать хозяина о прогрессе в работе -AddHandler Worker.Log, AddressOf LogHandler. Создаем новый поток, скидываем ссылку на него в менеджер потоков (чтобы знал, за кем присматривать) и запускаем поток.

```
Dim t As New Thread(AddressOf Worker.Start)
TM.Threads.Add(t)
t.Start()
```

И в отдельном потоке запускаем менеджер потоков:

```
Dim TMThread As New Thread(AddressOf TM.JoinThreads)
TMThread.Start()
```

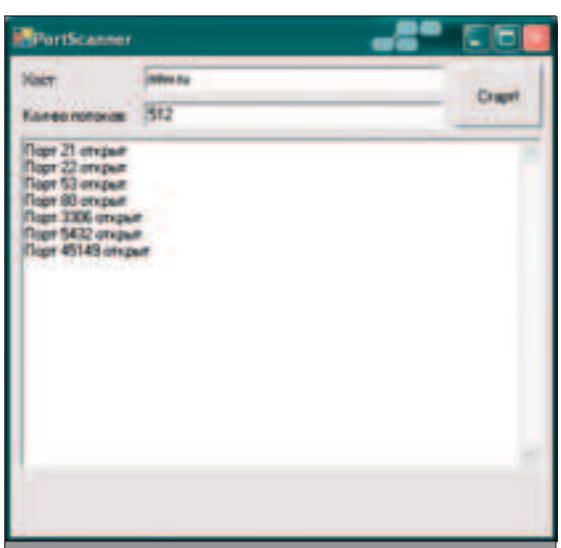
Также в frmMain определены две процедуры, обработчики событий Log и Done. Их не обязательно комментировать.

```
Public Sub LogHandler(ByVal Port As Int32, ByVal Opened
As Boolean)
SyncLock Me
PortsList(Port) = Opened
PortsScanned += 1
lblProgress1.Text = "Отсканировано " &
PortsScanned.ToString & " портов"
lblProgress2.Text = "Сканируется порт " & Port.ToString
End SyncLock
End Sub

Public Sub DoneHandler()
Dim sb As New System.Text.StringBuilder
For i As Int32 = 1 To 65536
If PortsList(i) = True Then
sb.Append("Порт " & i.ToString & " открыт" &
ControlChars.CrLf)
End If
Next
txtLog.Text = sb.ToString
MessageBox.Show("Отчет готов!")
End Sub
```

Теперь вернемся к менеджеру потоков. Всегда менеджерам жилось лучше, чем рабочим :). Вот и наш менеджер только и делает что следит, отработали ли рабочие свою норму, а в конце рабочего дня рапортует боссу, что работа окончена.

```
Imports System.Threading
Public Class clsThreadManager
Public Threads As New ArrayList
Public Event Done()
Public Sub JoinThreads()
For Each t As Thread In Threads
t.Join()
Next
RaiseEvent Done()
End Sub
End Class
```

Вот, собственно, и все. Теперь, надеемся, ты имеешь хотя бы небольшое представление о технологии dot.NET. Начинать пользоваться ею или нет - конечно, личное дело программиста. Но мы на нее перешли и теперь нисколько об этом не жалеем (хотя все зависит от поставленных перед программистом задач). Полный текст программы ты сможешь найти на диске (а также тебя там ждет бонус - NET Framework, который теперь не придется качать).



Вот что получилось

BUCKSWARE

6th (6th@mail.ru)

ВЫДЕЛКА ШКУРОК В ДОМАШНИХ УСЛОВИЯХ

ПИШЕМ ПЛЕЕР С ПОДДЕРЖКОЙ СКИНОВ

Время диктует свои условия: если хочешь поднять спрос на свой продукт, сделай его привлекательным. Продавцы реального мира заворачивают товар в красочную обертку, а продавцы виртуального - работают над интерфейсом, способным заставить пользователя потерять голову и побежать покупать программу :). Одному из методов создания такого интерфейса - шкурам (скинам) - и посвящена эта статья.

Пользователь сегодня привередливый пошел, мало одной функциональности - красивый интерфейс требуют. Показываешь заказчику БД: клиент-сервер, резервное копирование, импорт/экспорт данных, шлюз в 1С, а он смену фона и шрифта таблиц просит :). Хотя понять его можно: ему предстоит целыми днями работать в конторе с твоим творением. Хочется эстетики, удобства, да и вообще, кто красивые вещи не любит? Вот и получается, что при продаже программы, будь то shareware или работа на заказ, профессионально сделанный и красивый интерфейс занимает далеко не последнее место. Теперь, когда software engineering - это настоящий рынок, приходится и методы использовать рыночные, то есть упаковывать товар красиво и со вкусом. Пользователь видит симпатичный сплэш во время загрузки, сменные скины, прилипание к краю экрана и прочие изыски и радуется - чувствует, что программа стоящая. Особенно важны скины. Дело в том, что как ни старайся, в плане интерфейса всем не угодишь, а тут человек сам решает, как будет выглядеть программа. Свобода выбора - ценная вещь, а тут она еще и реализуется просто. Конечно, если твое творение по количеству глюков обгоняет Win95 и с завидным постоянством предлагает аварийно завершиться, никакой красивый интерфейс тебя не спасет, но если программа работает, как старинные часы работы итальянского мастера, почему бы не добавить немного эстетики? Я убедил тебя освоить технологию скинов? Отлично, начинаем.

КОМПОНЕНТЫ

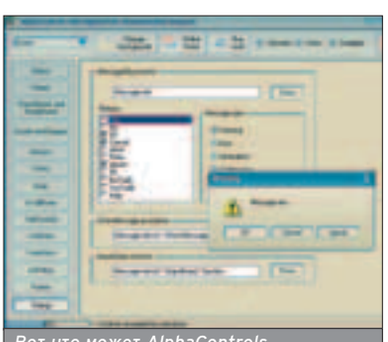
Если у тебя достаточно знаний, времени и энтузиазма для написания собственного skins-engine, можешь дальше не читать :). Однако предупреждаю: к тому времени, когда ты в одиночку напишешь нечто близкое к профессиональным скиновым движкам, весь остальной мир уже перейдет на Windows 2010 с полностью трехмерным интерфейсом. Есть известная притча "О том, как программисты льва в клетку сажали".

Пустыня. В ней лев. Клетка. Поставлена задача посадить льва в эту клетку. Программист на C++ проектирует клетку таким образом, чтобы лев был ее составной частью. При инициализации клетки лев автоматически генерируется внутри. Программист на Age говорит, что лев и клетка - это объекты разных типов и нечего морочить ему голову некорректными задачами. Программист на Delphi пишет во все конференции: "Народ, где взять компонент, который ищет в пустыне льва и помещает его в клетку?" :). Так что не будем забывать от стереотипе и начнем поиски компонента, способного облегчить поставленную задачу.

Библиотек, реализующих поддержку скинов, великое множество, но активно развивающихся и удобных в использовании гораздо меньше. Выберем для изучения библиотеку как раз из такого меньшинства: удобную, функциональную и не заброшенную программистом в дальний ящик.

AlphaControls (www.alphaskins.com) - честно говоря, не наш выбор. Сама библиотека довольно проста в использовании, но столь же просты ее возможности. Готовых скинов немного, утилиты для быстрого и легкого создания собственных шкур не наблюдается.

SUISkin (www.sunisoft.com/suiskin) - как утверждали создатели, "самый простой компонент для Delphi". И не соврали: чтобы включить поддержку скинов, достаточно просто кинуть на форму один компонент, существующие контролы "заскиновываются" автоматически. Очень удобно, если нужно добавить шкурки в уже написанную программу.



Вот что может AlphaControls

SUIPack (www.sunisoft.com/suipack) уже ближе к тому, что было бы приятно использовать. Продукт той же конторы, но помощнее. Активно развивается, прост в использовании, богат возможностями. Кроме вещей, связанных непосредственно со скинами, обладает возможностями для создания нестандартного интерфейса - непрямоугольных форм и т.п.

DynamicSkinForm (www.almdev.com) - регулярно обновляющийся проект, предлагает все то же, что и другие, плюс еще дюжину уникальных особенностей. Имеется SkinBuilder - программа, упрощающая создание собственных скинов, et cetera et cetera. Также Almediadev предлагает BusinessSkinForm - библиотеку для приложений, активно использующих различные таблицы, MDI и т.п.

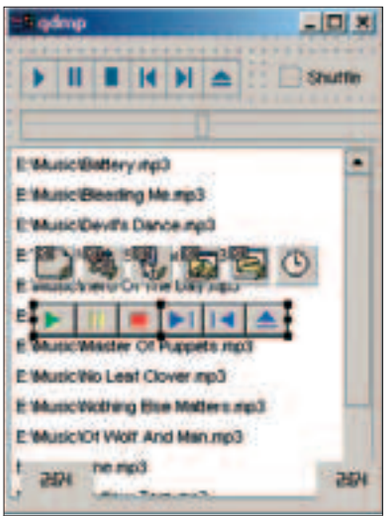
ThemeEngine (www.ksdev.com/themeengine) - бывший SkinEngine. Если бы не мое давнее пристрастие к DynamicSkinForm, выбрал бы именно эту библиотеку. Ничем не хуже, честное слово! Имеется ThemeWizard - утилита для натягивания скинов без изменения кода - и масса других, не менее удобных в использовании наворотов.

ПИШЕМ ПЛЕЕР

Не знаю как ты, но я при слове "скины" в первую очередь вспоминаю

Возможности SUIPack. Симпатично

Winamp. "Шкурный механизм" там реализован действительно очень качественно. Что-то на уровне WinAmp нам, конечно, с ходу не написать, но почему бы не попробовать? Как реализовать в своей программе поддержку скинов, написав простенькую программку - плеер? Я даже уже придумал ему название - QDMP (Quick & Dirty Media Player). Будем писать на Delphi, а в качестве skins-engine возьмем мой любимый DynamicSkinForm.



Даже не верится, что "это" когда-нибудь станет красивым

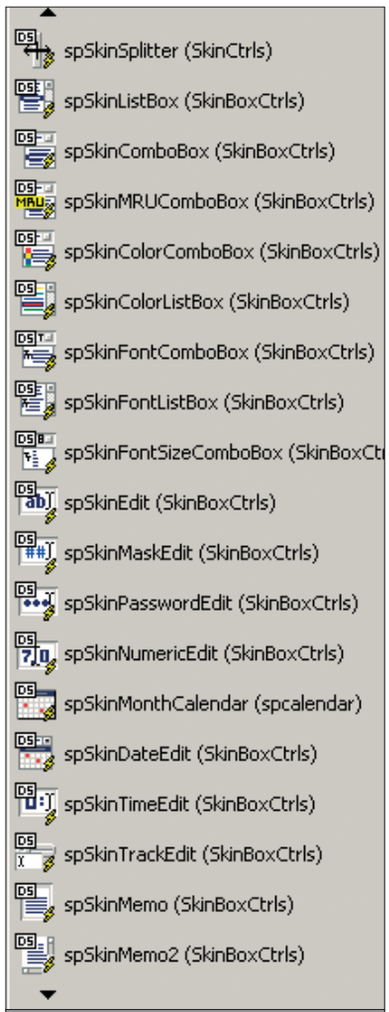
Прежде всего отправляйся на www.almdev.com/main/downloads.htm и качай библиотечку для своей версии Delphi. А если художественный талант у тебя, как и у меня, отсутствует в принципе, возьми там же базовую коллекцию скинов.

Библиотека устанавливается стандартно: распаковываем, в Delphi идем в Component->Install Packages, жмем Add и выбираем skinpackD7.bpl (для седьмой версии). После этого на палитре компонентов должны обосноваться две новые вкладки: Skin Pack и Skin Pack Dialogs. Можно приступать к программированию.

Первое, что надо сделать – поместить на форму spSkinData, который будет использоваться для открытия файла со шкуркой, а все остальные компоненты, в том числе форма, должны быть к нему привязаны.

На следующем шаге нужно поместить на форму компонент spDynamicSkinForm, который будет "натягивать" выбранный скин на программу. Его необходимо привязать к spSkinData через параметр SkinData. Если теперь запустить приложение, то все будет выглядеть весьма скромно, потому что spSkinData пока не ссылается ни на один скин. Ниже я объясню, как это поправить.

Раз уж взялись писать плеер, кроме компонентов шкурного механизма на форму нужно кинуть TMediaPlayer (у меня в сорцах – Player) и указать в его свойствах Visible=False. Кнопочки медиаплеера для "по умолчанию", конечно, ничего, но мы сделаем свои, поддерживающие шкурки. В принципе, можно натянуть скин и на стандартный TButton, но гораздо проще сразу воспользоваться готовым компонентом spSkinButton.



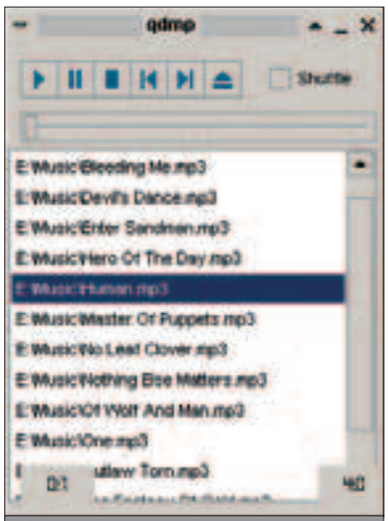
Это далеко не все компоненты DynamicSkinForm

Теперь необходимо разместить на форме шесть кнопок, которые должен иметь любой уважающий себя MP3-плеер: Play, Pause, Stop, Next, Forward и Open. Чтобы каждый элемент формы был в шкурке, каждый из них должен иметь параметр SkinData, связанный с spSkinData.

Кнопки кнопками, но какой плеер без плейлиста? SpListBox на форму! В программе мы не станем заморачиваться с форматированием вывода в плейлисте, а ограничимся добавлением туда полных имен файлов. Не очень эстетично, зато просто и быстро (основные достоинства Delphi - прим. ред.).

Натянуть шкурку на графические элементы формы очень просто, в DynamicSkinForm это делается всего одной строчкой. Правда, эта строчка должна выполняться в процессе загрузки программы, чтобы пользователь вдруг не увидел программу без шкурки ;).

```
procedure TfrMain.FormCreate(Sender: TObject);
begin
```



Выглядит пока скромно, но это временно

```
//загружаем скин из файла
spSkinData.LoadFromCompressedFile('BlueLight.skn');
//плеер - на начало плейлиста
Player.FileName := spListBox.Items[0];
Player.Open;
end;
```

DynamicSkinForm может хранить скины в нескольких форматах, но нам удобнее всего шкуры, содержащиеся в skn-файлах. В таком виде вся шкура содержится всего в одном файле, а так как файл сжат, он получается очень скромных размеров.

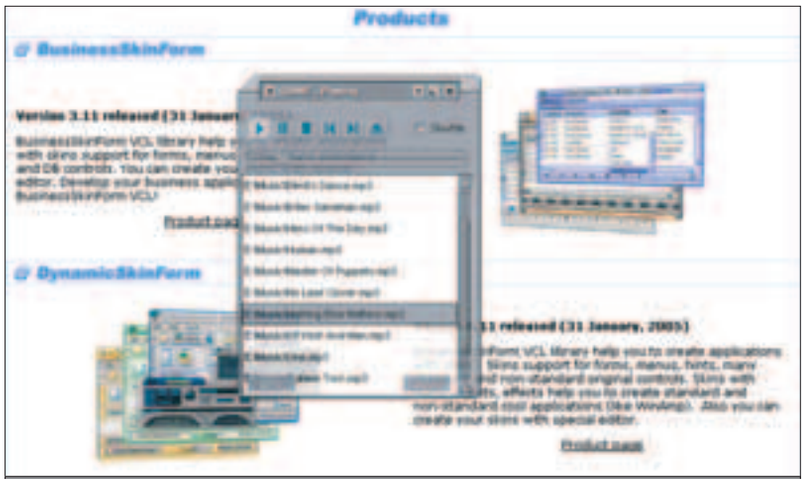
Удобно!

Поехали дальше. Было бы неплохо добавить возможность смены скинов прямо в процессе работы плеера, чтобы не только ты, но и пользователь мог выбрать понравившуюся тему. Реализуем это в виде всплывающей менюшки, для чего положим на форму spOpenSkinDialog, spSkinPopupMenu и создадим три пункта меню: Set new skin (для установки шкурки), Transparent (для установки прозрачности шкурки) и Exit (понятно для чего). Заставим это работать следующим образом:

```
procedure TfrMain.N1Click(Sender: TObject);
begin
//покажем окно выбора скина и загрузим его в
spSkinData
 if spOpenSkinDialog.Execute then

spSkinData.LoadFromCompressedFile(spOpenSkinDialog.F
ileName);
end;

procedure TfrMain.N2Click(Sender: TObject);
begin
 //True/false параметр AlphaBelnd задает прозрач-
ность
 N2.Checked := not N2.Checked;
 spDynamicSkinForm.AlphaBlend := N2.Checked;
end;

procedure TfrMain.N3Click(Sender: TObject);
begin
 Close;
end;
```

»

На нашем диске ты найдешь исходники плеера и несколько популярных скиновых движков.



Сайт Almediadev сквозь призму QDMP

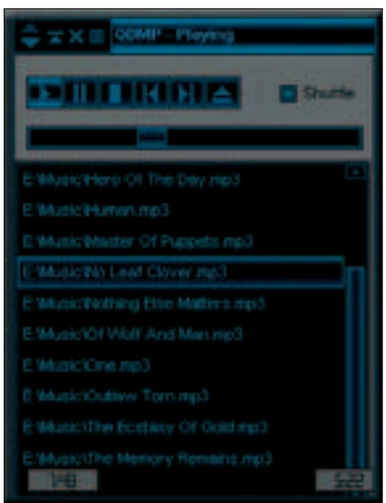
Теперь, стукнув по правой кнопке мыши, можно поменять шкурку у плеера или включить/выключить прозрачность. Это, конечно, здорово, но было решено написать плеер, а в нем одними скинами сыт не будешь. Нужно чтобы все кнопочки работали, а музыка играла, поэтому добавляем на форму spOpenDialog и пишем:

```
procedure TfrMain.spOpenBtnClick(Sender: TObject);
begin
 //добавляем выбранный файл в плейлист
 if spOpenDialog.Execute then
 spListBox.Items.Add(spOpenDialog.FileName);
end;
```

Теперь кидаем два spSkinLabel для отображения длины трека и текущего положения, а также spSkinTrackBar - для красоты. Каждый раз, когда в плейлисте будет выбираться новый файл, нужно будет выводить в лэйблы новые значения и сбрасывать trackbar в ноль. Чует мое сердце, эта последовательность встретится не раз, так что лучше вынести ее в отдельную процедуру:

```
procedure TfrMain.PlayFile(Filename : String);
begin
 //открываем указанный файл
 Player.FileName := Filename;
 Player.Open;
 //начинаем его проигрывать
 Player.Play;
 //настраиваем trackbar на длину трека в секундах
 //(TrackLength возвращает значение в миллисекундах)
 spTrackBar.MaxValue := Player.TrackLength[1] div 1000;
 //сбрасываем его в ноль
 spTrackBar.Value := 0;
 //выводим длину трека в удобоваримом виде
 spLenLabel.Caption := IntToStr(spTrackBar.MaxValue div 60) + ':' + IntToStr(spTrackBar.MaxValue - (spTrackBar.MaxValue div 60)*60);
end;
```

Осталось заставить все это работать в процессе воспроизведения музыки. Здесь все просто: помогут spTrackBar.Value и Player.Position (смотри полный исходник на диске). Теперь о плейлисте. Щелкнув по



Готово! :)

файлу в нем, нужно заставить этот файл проигрываться:

```
procedure TfrMain.spListBoxListBoxClick(Sender: TObject);
begin
 PlayFile(spListBox.Items[spListBox.ItemIndex]);
end;
```

Теперь нужно заставить нормально работать пять оставшихся кнопок:

```
procedure TfrMain.spPlayBtnClick(Sender: TObject);
begin
 Player.Play;
end;
//аналогично для Pause и Stop
procedure TfrMain.spPrevBtnClick(Sender: TObject);
begin
 //кнопка Previous должна переводить указатель на предыдущий
 //файл в плейлисте...
 spListBox.Selected[spListBox.ItemIndex-1] := True;
 //...и начинать его проигрывать
 PlayFile(spListBox.Items[spListBox.ItemIndex]);
end;
//аналогично для кнопки Next
```

ЧТО БЫ ЕЩЕ ДОБАВИТЬ?

Весьма к месту пришлась бы возможность выбора положения в файле с помощью trackbar'a. Опять же достаточно немного поиграть с величинами Player.Position и spTrackBar.Value. Ну и, конечно, какой плеер без shuffle'a! Кидаем spSkinCheckRadioBox и правим код кнопки Next:

```
procedure TfrMain.spNextBtnClick(Sender: TObject);
begin
 //если shuffle включен
 if spShuffle.Checked then begin
 //перейдем на случайный трек
 Randomize;
 spListBox.Selected[Random(spListBox.Items.Count)] := True
 end
//иначе все по-старому
else
 spListBox.Selected[spListBox.ItemIndex+1] := True;
 PlayFile(spListBox.Items[spListBox.ItemIndex]);
end;
```

И наконец, еще две вещи: строка состояния в заголовке формы и прилипание к краю экрана. Строка состояния позволяет приблизиться на один шаг к великому и ужасному Winamp'у, ну а прилипание давно стало стандартом де-факто в мире MP3-плееров и не только. С первым все просто:

```
const
//массив возможных состояний плеера
Modes: array[TMPModes] of string = ('Not ready', 'Stopped', 'Playing', 'Recording', 'Seeking', 'Paused', 'Open');

//OnNotify случается, когда режим работы TMediaPlayer изменяется
procedure TfrMain.PlayerNotify(Sender: TObject);
begin
 with Sender as TMediaPlayer do
 begin

 frMain.Caption := 'QDMP - ' + Modes[Mode];
 Notify := True;
 end;
end;
```

Прилипания - это вообще классика :). Добавляем в объявления:

```
procedure WMWINDOWPOSCHANGING(Var Msg: TWMWINDOWPOSCHANGING);
 message WM_WINDOWPOSCHANGING;
```

И пишем обработчик:

```
//выполняется при изменении положения окна
procedure TfrMain.WMWINDOWPOSCHANGING(var Msg:TWMWINDOWPOSCHANGING);
var
 Screen : TRect;
 StickAt : Word;
begin
//как близко нужно приблизиться к краю экрана
StickAt := 20;
//получаем размер экрана
SystemParametersInfo(SPI_GETWORKAREA, 0, @Screen, 0);
 with Screen, Msg.WindowPos^ do begin
                //корректируем положение окна
                Right := Right - cx;
                Bottom := Bottom - cy;
                if abs(Left - x) <= StickAt then
                                x := Left;
```

Какой бы выбрать?

Конечно, можно прочитать Help и клепать скины самостоятельно, но SkinBuilder здорово упрощает и ускоряет этот процесс.

```
            if abs(Right - x) <= StickAt then
    x := Right;
            if abs(Top - y) <= StickAt then
    y := Top;
            if abs(Bottom - y) <= StickAt then
    y := Bottom;
 end;
end;
```

Все, готово. Можно запускать и наслаждаться. Конечно, до нормального плеера еще далеко, но все-таки в первую очередь мы стремились ковать дизайн, а не mediaplayer :).

SKINBUILDER

Если тебя не устраивают стандартные скины Almediadev, не отчаивайся. На их сайте лежит крайне полезный пятиметровый файлик, включающий в себя справку по DynamicSkinForm, несколько примеров и утилитка с немудреным названием SkinBuilder. Эта программка и решит твои проблемы. Конечно, можно прочитать Help и клепать скины самостоятельно, но SkinBuilder здорово упрощает и ускоряет этот процесс. Работать с программой довольно просто, можно разобраться даже без чтения справки. Генератор скинов от "Алмедии" позволяет редактировать существующие шкуры, оценивать, как будет выглядеть форма и ее элементы, не выходя из программы и, естественно, создавать новые скины. После того как ты нарисовал кнопочки и окошечки в любимом редакторе, столкнешься с проблемой формирования их в виде, понятном DynamicSkinForm, и здесь SkinBuilder – неоценимый помощник. В общем, разбирайся и Photoshop тебе в руки. Может, через некоторое время твои работы попадут в коллекцию официальных скинов.

А как тебе такой цвет?

НАПОСЛЕДОК

Я уже говорил, что ни один скин в мире не спасет глючную программу от фиаско. В последнее время я часто наблюдаю, как люди, заполучившие такое мощное средство создания интерфейса, как, например, DynamicSkinForm, начинали забивать на основные принципы построения GUI, а от внешнего вида их программ меня начинает трясти. Ничто не поможет человеку приобрести вкус и чувство меры, так что я скажу одно: если твоя софтина спроектирована наспех и выглядит коряво и безвкусно, а ты надеешься сгладить впечатление каким-нибудь skins-engine - не расстраивайся! Просто ты неудачник и ставить SUIPack еще не время :).

И еще одно добавление к способам реального повышения рейтинга твоего творения. Если программа большая и ресурсоемкая, будет довольно утомительно ждать, пока она запустится, а при этом слушать тормозящий Winamp и тоскливо водить курсором по экрану. Приятная заставка (сплэш) ощутимо скрашивает ожидание и к тому же придает программе гораздо более профессиональный вид. Как это сделать? Элементарно!

Создаем новую форму, убираем рамку, кладем Image, присваиваем свойству Align значение alClient, загружаем картинку с заставкой. Теперь в Project->Options ставим эту форму в список Available Forms. Заставить все это работать довольно просто, в коде проекта после begin пишем:

```
try
//создаем нашу форму с заставкой
frSplash := TfrSplash.Create(Application);
frSplash.Show;
frSplash.Update;
//держим ее некоторое время
repeat
Application.ProcessMessages;
until frSplash.CloseQuery;
//убиваем ее и продолжаем работу
frSplash.Hide;
finally
frSplash.Free;
end;
// дальше все без изменений
```

Но это еще не все. Если программа загружается быстро, а похвастаться сплэшем все-таки хочется, заставим форму держаться на экране хотя бы несколько секунд. Кидаем Timer и ставим интервал на две-три секунды:

```
procedure TfrSplash.Timer1Timer(Sender: TObject);
begin
 //через две секунды таймер выключится и...
 Timer1.Enabled := False;
end;

procedure TfrSplash.FormCloseQuery(Sender: TObject;
var CanClose: Boolean);
begin
 //...мы разрешаем закрыть форму
 CanClose := not Timer1.Enabled;
end;
```

Также можно добавить код, закрывающий сплэш по щелчку или нажатию клавиши.

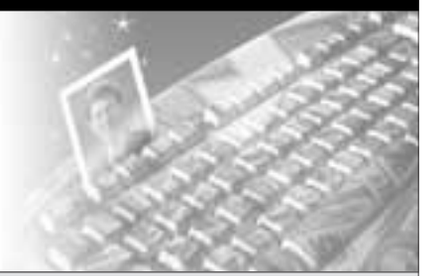
КУДА ПОЙТИ?

Сайты производителей ты найдешь в начале статьи, astalavista.box.sk. Адрес, я думаю, знакомый, так что проблем не будет. Напоследок хотелось бы привести адрес прекрасного форума по Delphi, где тебя всегда направят на путь истинный: www.delphimaster.ru. Если что-то непонятно, мой e-mail ты знаешь :). Удачи в создании настоящего X-интерфейса!

P.S: Пока я писал эту статью, Almediadev успели выпустить очередную версию DynamicSkinForm – 6.12. Вот тебе образец быстро развивающегося проекта :).

Степан Ильин aka Step (step@real.xakep.ru)

КОПИЛКА ТЕХНОЛОГИЙ



ЯЗЫК XML В РАЗРЕЗЕ

Каждый знает, что есть такой язык - XML, но мало кто может объяснить, что он представляет собой и для чего предназначен. Эта технология поражает количеством своих возможностей, благодаря которым ее можно применять почти повсеместно.

ПОЛЯРНОСТЬ ПРИНЦИПОВ

Возможно, кто-то заметит: "XML - это продвинутая версия HTML, верно?". Не совсем так. Это заблуждение очень распространенное, и виновато в его возникновении внешнее сходство XML- и HTML-документов. Обе технологии описаны с помощью обобщенного языка разметки SGML (Standart Generalised Markup Language), благодаря чему имеют схожий синтаксис, основанный на тегах (словах, заключенные в < и >) и атрибутах имя="значение". Еще одно сходство технологий - их активное использование в интернете, что еще сильнее вводит в заблуждение неподготовленного пользователя. Но! XML и HTML – это все-таки совершенно разные технологии.

Вспомним HTML – язык гипертекстовой разметки. Его основная задача состоит в описании внешнего вида web-страницы (расположение таблиц, настройки стилей, шрифта и т.п.), которое в соответствии со спецификацией стандарта обрабатывается браузером и выводится пользователю на экран. Расширенный язык разметки XML (eXtensible Markup Language) - напротив, не имеет какого-либо отношения к внешнему оформлению данных, зато напрямую определяет их структуру. В то же время XML требует, чтобы создание документа производилось строго в соответствии с описанной им структурой. Важно, что визуализация данных возложена на плечи сторонних средств, которые лишь обрабатывают XML-файл. Однажды созданный XML-документ может быть представлен в совершенно разном виде. Для примера возьмем наш журнал и опишем его в XML: зададим его структуру, содержащую название каждой статьи, вступление, непосредственно ее текст. Размеченный таким образом XML-документ может быть представлен самыми разными способами. Причем XML в этом случае выступает в качестве контейнера, хранящего строго структурированные данные, в то время как сам вывод информации осуществляется, например, с помощью того же HTML.

Подобный подход имеет массу достоинств. Например, если потребуется составить содержание номера, то из XML следует взять только заголовки. Верстаем журнал - придется выводить абсолютно всю информацию.

В конце концов, HTML – это независимый и полностью самостоятельный язык, а XML часто рассматривают как комплекс технологий. Другими словами, под аббревиатурой XML подразумевается не только расширенный язык разметки, но еще и десяток родственных технологий, таких как XHTML, XLS и т.д.

ВСКРЫТИЕ ПОКАЖЕТ

Существенные отличия между технологиями становятся еще более очевидными после тщательного осмотра. Используемые для составления HTML-документов теги и сама структура HTML предопределены, то есть программист во время составления HTML-документа может использовать только те теги, которые предусмотрены стандартом - например, <p>, <h1> и т.д.

В расширенном языке разметки используется совершенно другой подход. Программист XML может самостоятельно определять теги и значения, а затем с их помощью обозначать общую структуру документа. Итак, тег <p> в XML совершенно не обозначает "параграф", как это было бы в HTML. В зависимости от выбора программиста это может быть и какой-то параметр (parameter) или, например, счетчик страниц (page). Естественно, такой подход предоставляет практически неограниченные возможности для описания структуры данных. Приведу несколько примеров, чтобы точнее изложить свои мысли. Для начала пример кода HTML:

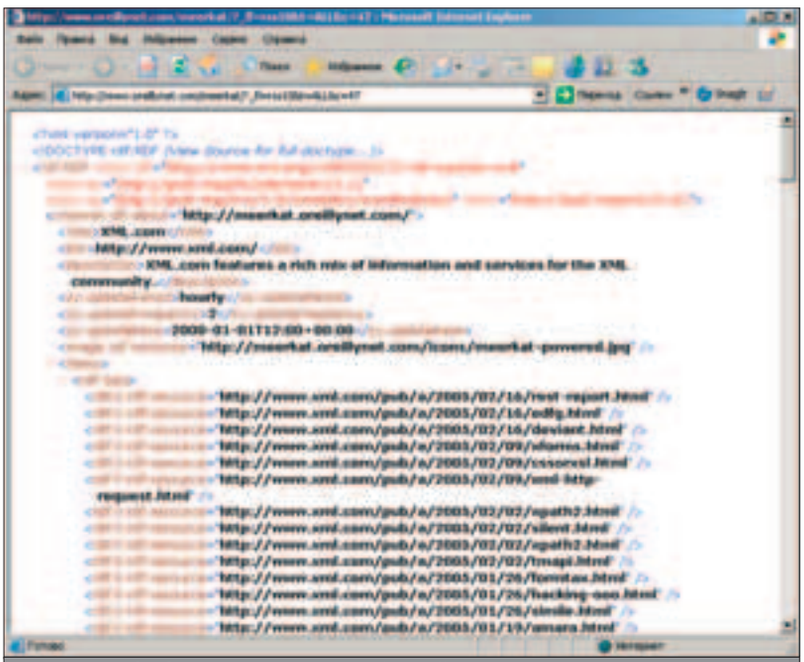
```
<H1>Контакты автора</H1>
<H2>Имя: Степан Ильин</H2>
<P>email: step@gameland.ru</P>
<P>icq: 111111</P>
```

Этот HTML-код представляет собой самую обыкновенную таблицу с координатами конкретного человека. Как ни крути, такое описание годится только для отображения небольшого количества информации. Статическая таблица с какой-нибудь тысячью записей - прямо скажем, убогая картина, потому что, кроме низкой гибкости, программист сталкивается с проблемой поиска информации. Несмотря на строгую структуру, парсер (обработчик документа) HTML в ходе анализа вполне может столкнуться с некоторыми неоднозначностями и в результате неправильно обработать код. Теперь посмотрим, как похожая, но несколько расширенная структура могла бы выглядеть в формате XML:

```
<?xml version="1.1"?>
<authors>
  <author>
    <first_name>Степан</first_name>
    <last_name>Ильин</last_name>
    <email>step@gameland.ru</email>
    <icq>111111</icq>
    <adress>
      <country>Россия</country>
      <city>Калуга</city>
      <street>Ф.Энгельса</street>
      <zip>248023</zip>
    </address>
  </author>
</authors>
```

Кусок исходного кода простейшего XML-парсера

Результат: вся информация логически разбита, что значительно упрощает процесс ее понимания и обработки. Ты в полной мере можешь указать, что представляет собой и где находится каждое из полей данных, значит, любой программный обработчик без труда разберется с ней, не полагаясь на интеллектуальный анализ, как в случае с HTML.



RSS-фиды, обработанные стандартным Microsoft'овским парсером

ПАРСЕРЫ XML

Как работают парсеры? Есть два современных подхода к анализу XML-файлов, каждый из которых по-своему хорош в определенных ситуациях. Первый использует так называемую модель DOM (Document Object Model). Ее суть в том, что парсер предварительно анализирует XML-документ и сохраняет полученное дерево в оперативной памяти. Далее, перемещаясь по полученному дереву, можно извлечь любую информацию. Этот способ очень простой, быстрый, но для работы с объемными XML-документами неуместен, так как предъявляет слишком высокие требования к количеству оперативной памяти. В таких случаях обычно используют другой тип парсеров, в основе которых применение SAX-библиотек (SAX - Simple API for XML). SAX-парсеры основаны на работе с событиями, то есть они последовательно просматривают весь документ и в случае обнаружения события (встретился определенный элемент, атрибут, значение атрибута) реагируют заданным образом (чаще всего извлекают данные). Ориентированность на события имеет ряд преимуществ: в первую очередь выигрыш в объеме потребляемых ресурсов, а часто еще и в быстродействии. Однако SAX-библиотеки имеют ограничения по структуре XML-файлов и в некоторых ситуациях физически неприменимы.

Стандартный Microsoft'овский парсер - **msdn.microsoft.com/XML/XMLDownloads**.
Все о XML-парсерах на PHP - **www.php.net/xml**.
Работа с XML на Perl'е - **www.xml.com/pub/a/2000/04/05/feature**.
Парсер XML на C++ - **http://xml.apache.org/xerces-c**.
То же самое, но для Java - **http://xml.apache.org/xerces-j**.
XML-модуль для Delphi/CBuilder - **www.icom-dv.de/products/xml_tools**.
Хороший исходный файл на Python'е - **http://uche.ogbuji.net/tech/4Suite/amara**.

XML – ЭТО ПРОСТО

Каждый документ в формате XML содержит сочетание разметки и текстовых данных. Разметка определяет логическую структуру документа, текстовые данные - хранимую информацию. Не нужно быть программистом, чтобы работать с XML. Язык вполне логичный и имеет всего несколько строгих правил. Главное - соблюдать их. Опишу некоторые особенно важные.

❶. На каждый открывающий XML-тег должен обязательно приходиться один закрывающий аналог. И если в HTML вполне допустимо и даже принято использования следующего кода:

```
<p>Новый параграф
<p>А это еще один параграф
```

То в XML подобное неприемлемо, поэтому код должен выглядеть следующим образом:

```
<p>Новый параграф</p>
<p>А это еще один параграф</p>
```

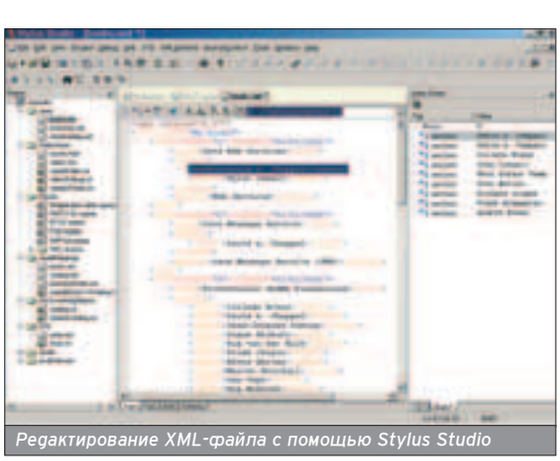
Внимательный читатель, возможно, возмутится и скажет, что в предыдущих примерах тег <?xml..> не имеет закрывающего аналога. Это не ошибка, скорее исключение! Тег имеет вид <?xml version="n.n"?> и декларирует использование n.n-версии XML. На момент написания статьи последней модификацией XML была 1.1. Однако 1.0 по-прежнему широко используется, да и различия даже между этими версиями весьма несущественные.

❷. Полужирный, курсивный текст
Этот HTML-код, в котором теги накладываются друг на друга (нарушается порядок вложенности), хоть и не приветствуется, но обрабатывается большинством браузеров. Чего не скажешь об XML: синтаксис здесь намного строже и такие накладки недопустимы. Правильный XML-код должен выглядеть следующим образом:
Полужирный, курсивный текст

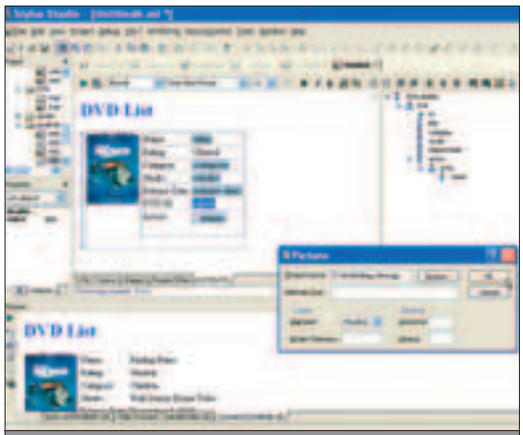
❸. XML поддерживает многоуровневое вложение тегов. По правде говоря, это даже поощряется как способ »

Более детальную информацию об XML ищи на сайте Web-консорциума - www.w3c.org, а толковые мануалы - на www.w3schools.com.

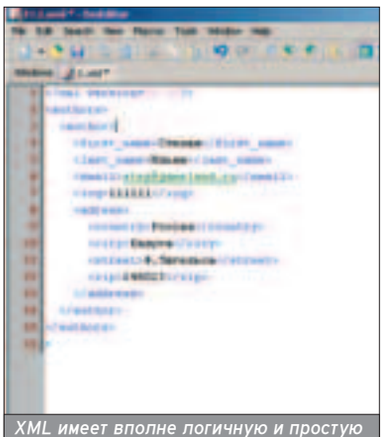
Проверить формальную правильность синтаксиса HTML- и XHTML-документов, а также XML-схем можно с помощью специальных валидаторов. Наиболее популярным и авторитетным является validator.w3.org.



Редактирование XML-файла с помощью Stylus Studio



Был XML - стал HTML



XML имеет вполне логичную и простую структуру

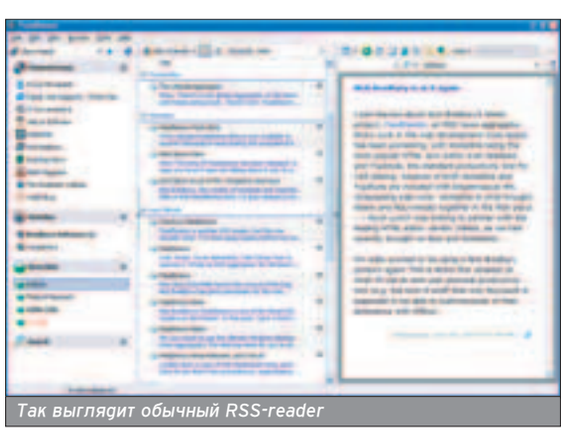
Формат XML требует строгого соблюдения всех заданных спецификацией правил.

задания иерархической структуры данных. Из приведенного выше примера видно, что такие элементы, как <first_name> и <email>, используются для хранения информации, в то же время тег <address> фигурирует исключительно в целях указания нового уровня иерархии. Главное здесь - учитывать, что на самом верхнем уровне всегда должен находиться только корневой элемент. В нашем случае таким элементом является <authors>. В противном случае XML-файл считается неправильным.

❶. В отличие от HTML, XML-теги зависят от регистра написания. Так, тег <Spec> не является тем же самым, что и <spec>. Открывая и закрывая теги, нужно внимательно следить за тем, чтобы они имели один и тот же регистр.

❶. Теги, обозначающие какой-либо элемент, хоть и являются основными, отнюдь не единственные. Более того, любым элементам могут быть присвоены один или несколько характеризующих их атрибутов. Эта возможность полностью аналогична имеющейся в HTML, где, к примеру, элементу <table> можно присвоить атрибут align="left". Учти, что любые значения атрибутов в XML в строго обязательном порядке должны быть заключены в кавычки.

WML (Wireless Markup Language) – язык для создания WAP-сайтов, которые можно просматривать с помощью любого мобильного телефона. Эта спецификация полностью основана на XML и определяет синтаксис, переменные и элементы, используемые в файлах WML.



Так выглядит обычный RSS-reader

КАК С ЭТИМ РАБОТАТЬ?

Разумеется, я затронул лишь малую часть всех правил и конструкции языка. На самом деле их намного больше. Но даже этих знаний достаточно, чтобы эффективно и полноценно использовать XML. Напрашивается вопрос: а как, собственно, использовать? Для этого как минимум понадобится так называемый парсер, который последовательно обрабатывает XML-файл и выделяет хранимую в нем информацию. Он может быть программой или просто программным модулем с такими же функциями. Так или иначе, для работы с XML необходимо использовать парсеры всегда и везде. Как для банального отображения содержания документа в HTML-файле (при этом полученные данные необходимо привести в удобопонятный для браузера вид), так и для сложного поиска конкретной записи. В последнем случае ко всему прочему требуется проверять соответствие текущей записи заданными условиям и параметрам.

Формат XML требует скрупулезного соблюдения всех заданных спецификацией правил. Парсеры учитывают такую жесткость языка и поэтому не допускают каких-либо отклонений. Любое семантическое нарушение они воспринимают как ошибку.

Важно, что XML-документ проходит проверку по нескольким параметрам. Первая – на семантическую правильность (все ли теги закрыты, проверка на запрещенные символы, перекрытие тегов и т.д.). Вторая – на соответствие хранимых данных типу, обозначенному структурой документа. До сих пор мы считали, что хранимые данные могут иметь произвольное значение и любой тип данных. Это не совсем так. Дело в том, что все предыдущие XML-файлы обозначают логическую структуру документа, но при этом не накладывают ограничений на используемые данные и их атрибуты. Такие документы называются формально правильными и, по правде говоря, в последние время стали особенно широко распространенными. В то же время средствами XML вполне можно описать конкретные правила формирования элементов, типы полей и т.д. Тип элемента может задаваться двумя способами: с использованием DTD (Document Type Definition) или более гибкого современного средства - языка схем XSD (XML Schema Definition). DTD - не самое удобное средство. Громоздкие описания данных поначалу игнорировались большинством парсеров, и спецификация не получила широкого распространения. Язык схем XSD в свою очередь с самого начала разрабатывался для замены DTD и поэтому встроен в XML. По сути, создаваемая для какого-либо документа схема сама является XML-документом, так что может быть обработана любым парсером.

РАСШИРЕНИЯ XML

Важной особенностью XML является его расширяемость. Наряду с самими XML активно используются родственные спецификации, которые предлагают целый ряд дополнительных возможностей, которые во многом дополняют XML и таким образом способствуют его повсеместному внедрению.

Возьмем, к примеру, упомянутый мной XML-документ, содержащий информацию об авторах Спеца. По сути, это текстовая база данных, которую при желании можно преобразовать и представить в удобном для пользователя виде. Реализовать это несложно. Первое, что приходит в голову - обработать XML с помощью парсера, а затем, используя каскадные таблицы стилей (Cascading Style Sheet, CSS), оформить информацию в виде таблицы HTML. Однако это не самое изящное решение. В большинстве случаев предпочтительнее использовать вспомогательное дополнение XML - расширяемый язык стилей XSL (eXtensible Stylesheet Language). Базовой доктриной XSL является то, что для каждого документа разработчик вправе задать правило, определяющее вид его отображений на экране. XSL-процессор обрабатывает данные в формате XML и таблицу стилей в формате XSL, после чего создает отображение данных в соответствии с заданными стилями. Короче говоря, XSL в первую очередь предназначен для визуального представления XML-документов. С ее помощью можно также сортировать данные, производить по ним поиск, удалять или добавлять прямо из браузера. А использование ее подразделения XSLT в зна-

www.stylusstudio.com – сайт отличной программы для работы с XML

чительной мере упрощает и ускоряет процесс преобразования XML-документа в другие форматы.

Для отображения данных XML-документа совсем не обязательно преобразовывать их именно в HTML. В связи с широким и быстрым распространением XML Web-консорциум (www.w3c.org) разработал модификацию HTML. В результате с января 2000 года язык расширенной гипертекстовой разметки XHTML (EXtensible HyperText Markup Language) официально рекомендуется всем разработчикам для оформления web-документов. Во многом он практически идентичен своему предшественнику (то есть перейти на него совсем несложно), но в тоже время более практичен и функционален. Приоритетные направления – изящное взаимодействие с XML и более жесткие правила по отношению к синтаксису (вспоминаем требования XML).

Во время разработки крупных проектов описанием структуры только документа не отделаешься. XML предоставляет широкие возможности создания новых описаний при комбинировании уже имеющихся наработок, но все это с некоторыми трудностями. Рассмотрим пример: пусть XML-документ имеет теги <head> и <body>, которые соответственно обозначают тело и туловище человека. Теперь представь, что в этот же XML-документ необходимо вставить небольшой кусок кода XHTML, который в свою очередь также имеет теги <head> и <body>, предназначенные для выполнения разметки. Вопрос: как отделить теги основного XML-документа от тегов XHTML-вставки? XML поддерживает механизм пространства имен и поэтому спасет тебя от такой путаницы. Этот механизм предлагает использовать для каждой схемы собственный уникальный префикс, однозначно определяющий значение конкретного тега. Для того чтобы устранить неоднозначность, программисту достаточно правильно объявить пространства имен и в зависимости от ситуации добавлять к каждому тегу соответствующий префикс.

Ты уже убедился, что технология XML - это растущее множество модулей, предоставляющих полезные функции для решения самых разнообразных задач. К уже перечисленным дополнениям можно добавить спецификацию расширяемого языка указателей (XPointer), которая определяет способы указания и ссылки на конкретные фрагменты XML-документа. В принципе XPointer немного похож на URL, но вместо указания на документы в Сети обращается к фрагментам данных внутри XML-документа.

А, например, спецификация XLink определяет общую структуру для выражения гиперссылок в XML-документах. XInclude – новый механизм для утонченного объединения XML-файлов. Этот список можно продолжать и продолжать.

Используем XML

К главным достоинствам XML я всегда относил его универсальность, благодаря которой он и стал таким популярным. Язык имеет множество вариантов применения - предлагаю рассмотреть некоторые из них. В первую очередь XML – это, конечно же, средство описания структуры документа. Язык, как никакой другой, подходит для определения практически любых структур и типов документов в обычном тестовом виде. Несмотря на внешнюю простоту, XML обладает эффективными механизмами контроля правильности данных, позволяет производить проверку иерархических отношений внутри документа. XML-схемы позволяют описать практически любой документ, будь то документация к простенькой программе или широко разветвленный web-сайт.

Само описание информации является другим аспектом использования XML - хранение данных. Действительно, XML можно смело рассматривать как контейнер, содержащий самую разнообразную информацию. Особенно если учитывать, что многие современные СУБД поддерживают формат XML или, по крайней мере, умеют преобразовывать информацию в вид XML-документов. Я уже не говорю о специализированных XML-СУБД, таких как Tamino (www1.softwareag.com/Corporate/products/tamino). Здесь, правда, далеко не все программисты приветствуют хранение данных в текстовом формате. Главный их аргумент в том, что XML-файлы почти всегда имеют размер больше, чем размер данных, хранящихся в бинарном виде. Но не стоит забывать о том, что занимаемый информацией объем - не самый важный критерий выбора БД, особенно сейчас, когда дисковое сильно подешевело и стоит, по сути, копейки. Тем более что любая текстовая информация при необходимости может быть оперативно и быстро сжата самыми обычными архиваторами.

XML умеет не только хранить информацию, но и помогать в нелегком деле обмена ею. А почему бы и нет? Открытый и понятный стандарт, масса предназначенных для его обработки программ и открытых модулей для программистов - все только благоприятствует такому применению. Вспомни технологию RSS (Really Simple Syndication), которая является отличным механизмом для экспорта информации с web-сайта (к примеру, новостной ленты). Экспортируемые данные хранятся в специальном файле, который описывается языком XML и доступен для скачивания. В погоне за обновленной информацией рядовому пользователю совершенно не обязательно заходить на какой-то сайт - достаточно загрузить обновленный RSS-файл и обработать его специальным парсером (RSS-reader'ом). Или другой пример – сервис Яндекс. XML позволяет передавать автоматические поисковые запросы к Яндексу и публиковать его ответы прямо у себя на сайте в собственном дизайне. С его помощью можно без сложных скриптов наладить эффективный поиск по крупному сайту (правда, в той мере, в какой этот сайт проиндексирован Яндексом) с невероятным быстродействием. Лихо, правда?

Это еще не все! В последнее время XML нередко используют в качестве кроссплатформенного способа описания механизмов какого-либо скриптового языка. Один раз описав все необходимые инструкции, программист, к примеру, получает возможность сконструировать с помощью XML сложнейший конфигурационный файл с древовидной иерархической структурой. Благодаря этому отпадает всякая необходимость написания обработчика такого сложного конфига: исходные коды парсеров XML доступны на любом языке программирования. Более того, такой подход значительно удобнее, чем использование порядком устаревших INI-файлов, накладывающих ряд неприятных ограничений.

XML.org – здесь ты найдешь много интересного о применении XML

Content:

Ермолаев Евгений aka Saturn (saturn@linkin-park.ru)

БЕЗ «БУМАЖКИ» НИКУДА

СЕРТИФИКАЦИЯ И ОБРАЗОВАНИЕ ПРОГРАММИСТА

Чтобы пробиться на перспективную должность в любой организации, необходимо иметь хорошее образование. Сегодня высшее образование уже не может стать стопроцентной гарантией успешного трудоустройства, особенно это касается IT-специалистов. Дополнительные сертификаты, подтверждающие квалификацию в той или иной предметной области, повышают твои шансы на популярность среди работодателей.

Знания, которые предлагает классическое высшее образование, все меньше соответствуют реальным требованиям рынка труда индустрии информационных технологий. Такое положение дел в первую очередь связано со скоростью развития IT: эксперты признают, что и одного года достаточно, чтобы серьезно отстать от технологий. Чтобы спасти мир от беспощадности времени, крупные IT-компании стали вводить так называемые программы сертификации, о важности и престиже которых сегодня не говорит только ленивый. Бумажки с заклинаниями "MCSE", "CCIO", "MCP","CNE" действуют на IT-специалистов так же, как на кота валерьянка. Разберемся что к чему.

ЧТО ЭТО ТАКОЕ?

Профессиональная сертификация специалистов в области информационных технологий существует уже более пятнадцати лет, и появилась она благодаря гигантам IT-рынка, начальство которых заботится об увеличении прибыли, но и о полезном в этом деле профессионализме работников не забывает. В 1989 году Novell первой запускает программу Certified Novell Engineer (CNE). Уже к 1993 году около 20-ти IT-компаний предложили свои программы авторизованного обучения.

Сертификация - это процедура подтверждения профессиональных качеств специалиста путем сдачи им некоторых экзаменов. Если попытка была успешной, кандидату присваивается звание и в течение некоторого времени выдается именной международный сертификат. Чаще всего экзамен представляет собой тест на английском языке. Сдача экзамена, естественно, платная - цена на сертификационные тесты колеблется от $20 до 300 за тест. Для получения звания нужно сдать несколько тестов, входящих в так называемый трек. Сертификаты делятся на несколько категорий.

СЕРТИФИКАТЫ ДЛЯ РАЗРАБОТЧИКОВ

Рынок сертификации для разработчиков намного меньше, чем, например, для сетевых специалистов. Это обосновано довольно высокими требованиями к первоначальной подготовке специалистов. Даже на авторизованные курсы не стоит идти учиться не имея как минимум годового опыта профессионального программирования с использованием выбранного инструмента. Минимум, необходимый для сертификации программиста, - глубокие знания и обширный опыт работы. Однако несмотря на сложность получения, такой сертификат дает гораздо больше преимуществ. Программисты оплачиваются намного лучше,

Компания Novell первой начала сертификацию специалистов

Сертификат Borland теперь можно получить только в офисах компании

чем системные администраторы, такая работа менее нервная и более творческая, предоставляет гораздо больше возможностей для роста.

В отличие от сетевых специалистов, для которых сертификация постепенно переходит в разряд необходимых атрибутов, наличие звания у программиста - пока большая редкость. В США всего 20% программистов имеют хоть какую-то сертификацию. Стоит сказать, что интерес к сертифицированным специалистам наблюдается пока только за рубежом. К сожалению, в России подобный сертификат редко позволяет получить существенно более высокую зарплату. Однако жизнь не стоит на месте, и все идет к тому, что в скором времени сертифицированные специалисты станут нужны и у нас. Попытаемся рассмотреть наиболее престижные статусы, которые может получить программист.

BORLAND CERTIFICATION

■ Фирма Borland является производителем инструментов для разработчиков, поэтому появление сертификационной программы по поддержке ее продуктов было предсказуемым. Однако если изначально такие сертификаты можно было получить во многих центрах тестирования, то теперь это можно сделать только в офисах компании Borland. Поэтому доступность экзамена зависит еще и от степени удаленности офиса. Сегодня доступны несколько статусов Borland Certified Developer:

Borland App Server;
Borland Enterprise Server - AppServer Edition;
Borland Enterprise Server - VisiBroker Edition;
C++ Builder;
Delphi;
JBuilder;
Kylix.

IBM

■ Доступные статусы:

Certified Developer;
Certified Solution Developer;
Solutions Expert Enterprise Developer.

Стоимость экзамена довольно велика - от $125. Большинство статусов связано с продуктом WebSphere. До недавнего времени некоторые из экзаменов засчитывались при получении статусов JCert (консорциум поставщиков и производителей учебных курсов по Java). Однако JCert уже распался.

Программа Lotus хорошо развита

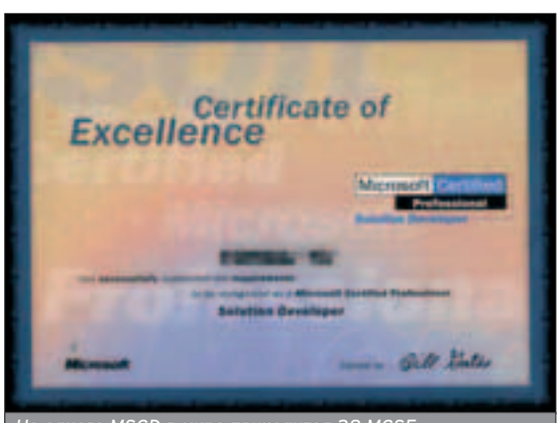
LOTUS

■ Доступные сертификаты: CLS, CLP, Principal CLP. Первый этап сертификации Lotus (подразделение IBM Certified Lotus Specialist, CLS). Для получения этого статуса достаточно сдать только один экзамен в любом из трех треков. Следующий этап - статус Certified Lotus Professional(CLP). Здесь нужно сдать два экзамена. Наконец, высший уровень сертификации - Principal CLP. Программа Lotus хорошо развита, предлагает углубленные учебные пособия и обновления статуса (при условии сдачи одного экзамена).

MICROSOFT

■ Одна из самых крупных компаний разработчиков ПО в настоящее время предлагает два статуса: Microsoft Certified Application Developer (MCAD) и Microsoft Certified Solution Developer (MCSD). Для получения статуса MCAD достаточно сдать три экзамена: два обязательных и один по выбору. Статус MCSD можно получить или по треку .NET, или по Visual Studio 6. В первом случае потребуется сдать пять экзаменов, во втором - только четыре. Статус MCSD (Microsoft Certified Solution Developer), пожалуй, самый престижный у Microsoft. Достаточно сказать, что на одного MCSD в мире приходится 28 сертифицированных инженеров (MCSE), а в России этот статус имеют около 1000 человек.

Статус MCSD подразумевает не только знание основ языков программирования, но и высокую образованность в деле комплексного подхода, создания инфраструктуры программных продуктов, знания и использования современных технологий программирования задач для Windows (COM, DCOM, ActiveX), web-технологий (ASP, DHTML, vb/java scripting) и концепций Client/Server и Multitier для созданий распределенных систем. В России сейчас отмечается устойчивый рост популярности этого статуса: увеличилось количество сдаваемых тестов по MS Visual C++, Visual Basic, Visual InterDev . "Русская Редакция" (серия "Учебный Курс") и Издательский дом "Питер" (серия "Сертификационный экзамен - экстерном") выпус- »



На одного MSCD в мире приходится 28 MCSE

тили практически весь спектр книг для подготовки к получению MCSD.

ORACLE

■ Доступные сертификаты: PL/SQL Developer Associate, Forms Developer Professional.

Чтобы получить статус Application Developer Certified Associate, претендент должен сдать два экзамена: введение в Oracle 9i и один по программированию на PL/SQL.

Также Oracle продолжает предлагать свой статус Oracle Forms Developer Certified Professional 6/6i. Популярный статус требует сдачи четырех экзаменов.

Ни один из этих статусов не требует обязательного прохождения сертифицированных курсов.

ЦЕННОСТЬ СЕРТИФИКАТОВ

■ Наиболее популярны на современном рынке сертификации Microsoft и Cisco Systems. Если первая добилась популярности за счет повсеместной рекламы, то вторая - как наиболее высокооплачиваемое направление. И если в компетенции специалиста с сертификатом Certified Cisco Internetwork Expert(CCIE) никто не сомневается (в мире около 6000 таких специалистов), то репутация Microsoft уже ставится под сомнение. С одной стороны, рекламной политикой компания добилась узнаваемости сочетаний типа MCAD или MCSE. С другой стороны, сертификаты Microsoft перестали быть чем-то сверхестественным, зато стали чем-то массовым (массовость и элитарность - вещи несовместимые). Конечно, предпринимаются меры для возвращения былого престижа "корочек", однако репутация уже серьезно подмочена.

Другая не менее важная проблема - появление так называемых "бумажных специалистов". Разного рода "шпаргалки", то есть списки правильных ответов на вопросы тестов, и тому подобные вещи способствуют размножению "бумажных специалистов" и снижению престижа того или иного сертификата.

Есть, однако, несколько званий, которые никогда не теряли своей престижности и высоко ценятся. Это звания, присваиваемые разработчикам: OCD (Oracle Certified Developer), SCPJP (Sun Certified Programmer for the Java Platform), IBM CD (IBM Certified Developer), IPC (Borland Product Certified), звания для сетевых специалистов от Novell, Cisco и др. Ради восстановления исторической справедливости скажу, что еще довольно высоко ценится MCSD (Microsoft Certified Solution Developer).

Во всем мире только 6000 специалистов имеют сертификат CCIE

КОЛИЧЕСТВО СЕРТИФИЦИРОВАННЫХ СПЕЦИАЛИСТОВ В МИРЕ

■ Количество сертифицированных специалистов Microsoft в мире:
MCP - 610 623;
MCSE - 391 007;
MCSD - 14 967;
MCDBA - 15 115.
Общее число сертифицированных специалистов Microsoft составляет 1 021 528 человек.

■ Количество сертифицированных специалистов Microsoft в России:
MCP - 6 933;
MCSE -3 076;
MCSD - 913;
MCDBA - 1 106.
Общее число специалистов по России и странам СНГ - 12 028.

■ Количество сертифицированных специалистов Novell в мире:
CNA - 400 000;
CDE - 120;
CNE - 200 000;
CIP - 2 500;
CNS - 16 000;
MCNE - 23 000;
CNI - 4 000;
Всего - 645 620.

■ Количество сертифицированных специалистов CCIE (Cisco Certified Internetwork Expert) в мире:
Америка - 2 746;
Европа, Средняя Азия и Аффрика - 1 725;
Россия - 54;
Тихоокеанский регион - 915;
Азия - 671;
Австралия - 202;
Новая Зеландия - 42;
Китай - 152;
Япония - 248;
Всего - 5432.

Данные на февраль-май 2001 года, по России - на февраль 2003 года (источник - **www.certification.ru**).

СТОИТ ЛИ ИГРА СВЕЧ?

■ На сегодняшний день в России еще очень мало работодателей, которые готовы оценить IT-специалиста, получившего сертификат, выше, чем его собратьев. С одной стороны, для успешного трудоустройства программисту требуется закончить престижный ВУЗ, иметь опыт работы и, желательно, получить сертификат от известной фирмы. Этот путь гарантирует престижную и перспективную должность практически в любой крупной компании. С другой стороны, возможен вариант, когда есть навыки, но нет высшего образования и разного рода "корочек". В этом случае нормальная должность может быть получена, если кто-нибудь "приметит" тебя как хорошего специалиста. Можно заняться написанием freeware-программ и размещать их на специальных сайтах. Можно участвовать в соревнованиях наподобие Софтулийских игр. И первый, и второй путь могут привести тебя к желаемому результату, однако чем больше у тебя доказательств своей компетентности в выбранной области, тем больше шансов на успех.

Так что дерзай - практически к любому средству разработки можно получить профессиональный сертификат. В любом случае, результатом получения сертификата будет как минимум красивая бумажка с надписью "Certified...".

СПЕЦИАЛЬНЫЙ ВЫПУСК!

СПЕЦИАЛЬНЫЙ ВЫПУСК В ПРОДАЖЕ С 13 АПРЕЛЯ

Подписка:
тел. 8-800-200-3-999
(звонок бесплатный)

- рецензии на фильмы (отечественные и зарубежные)
- оценка качества изображения и звучания
- информация о дополнительных материалах
- материал о военном кино
- экспорт российских фильмов о войне: интервью с компанией RUSCICO
- реставрация старых фильмов: интервью с КВО «Крупный план»

НА DVD-ПРИЛОЖЕНИИ
ФИЛЬМ
«В БОЙ ИДУТ ОДНИ СТАРИКИ»

ВТОРАЯ МИРОВАЯ ВОЙНА В КИНО

Крис Касперски aka мыщъх

ПРОГРАММИРОВАНИЕ ДЕНЕГ ЛОПАТАМИ

МНЕНИЕ СПЕЦА О ЗАРАБОТКЕ ДЕНЕГ НА КОДИНГЕ

Говорят, что наука - это удовлетворение собственного любопытства за чужой счет. Программирование, между прочим, - тоже! Графический интерфейс аляповатых приложений скрывает огромные денежные вложения, и если вовремя подсуетиться, можно кое-что отхватить от этого куска. Хочешь узнать, какова твоя рыночная стоимость и как не продешевить на продаже продуктов своего труда?

СЧАСТЬЕ НЕ В ДЕНЬГАХ, А В УМЕНИИ ЗАРАБАТЫВАТЬ ИХ

Стоит ли вообще стремиться к деньгам? Быть вовлеченным в безжалостный круговорот потребления и выделения. Зарабатывать и накапливать. Накапливать и тратить. Тратить и приближаться к иллюзорной картине коллективной галлюцинации, называемой "счастьем". Скажи честно - оно тебе надо? Для счастья не нужно ничего, кроме сознания, а деньги приносят только пустоту и страдание, но обычно это понимаешь лишь тогда, когда их (денег) становится слишком много и уже поздно что-либо менять. Жизнь прожита, а вложенное в деньги время не вернуть назад. Человек-потребитель в конечном счете живет и работает на унитаз. Может, настала пора остановится и подумать о чем-то более возвышенном?

Посмотри "Бойцовский Клуб" (Fight Club), найди и прочитай "Поколение П" Виктора Пелевина и "99 франков" Фредерика Бегбедера. Вставляет всерьез и надолго. Пошли карьеру, деньги, шмотки и займись, наконец, самим собой. Тем, что ты бы с удовольствием делал и бесплатно, тем, что наполняет жизнь смыслом, удерживает у монитора до самого утра, тем, что тебе по-настоящему интересно, а я покажу, как обернуть этот интерес в деньги. Никакого противоречия здесь нет. Плохо когда деньги становятся самоцелью, и хорошо если они средство (существования) или инструмент (реализации своих идей). Вот об инструментальных средствах и поговорим.

"Буду программировать на HTML за еду"

КУДА ПОДАТЬСЯ?

Нашим предкам было хорошо. Они сидели в пещерах, укрывались шкурами, добывали огонь трением, а топили распечатками от АЦПУ, находясь на полном обеспечении у государства, решавшего все кадровые вопросы за тебя. С приходом свободы, а значит, и возможности выбора, позиция "самых сильных программистов в мире" значительно пошатнулась. Уже никто не собирался оплачивать из своего кармана поиск недокументированных возможностей в недрах операционной системы или программирование в чистом машинном коде. Преемственность программистской школы нарушилась, старшие теперь перестали учить младших, преподаватели ВУЗов бесконечно далеки от проблем практического программирования и безнадежно отстают от прогресса.

Как достичь мастерства? Очень просто - устроится на постоянную работу в фирму, где еще сильны программистские традиции и имеются профессионалы, способные научить молодежь, передать ей часть своего опыта (только помни, что ремеслу не учатся - ремесло воруют). Трудовой договор (он же "контракт") бывает двух типов: ограниченный или бессрочный. В России чаще всего практикуется второй (устроился на работу, и не уволишь тебя, в "нормальных" фирмах легче терпеть безделье ничего не делающего человека, чем затевать волокиту с его увольнением, в общем, полная лафа: не так важно знать программирование, как законы).

Зарубежные фирмы в своей массе нанимают специалистов лишь на очень короткий срок (например, на год), после чего контракт может быть либо возобновлен, либо послан пинком под зад. Причины? Специалист не оправдал возложенного на него доверия или проект уже завершен и программистские услуги фирме больше не требуются. Тысячи специалистов, работавшие над "Боингом", были выброшены после того как птичка начала летать. Короче говоря, чем больше вкалываешь, тем быстрее тебя увольняют. Но на любую хитрость найдется свой прием. Умудренные опытом специалисты затягивают сдачу проекта всеми силами, каждый раз "находя" новый дефект, требующий доработки. Птиц вроде бы и летает, но в тоже время и нет. Работодатель нервничает, матерится, называет всех всякими нехорошими словами, но... вынужденно продлевает контракт.

Сейчас всюду требуют знания C++ и ActiveX/OLE, а как же те, кто любит асм? Что делать тем, кто живет в провинции, где в основном занимаются поддержкой и внедрением, а программирование задвинуто в глубокий подвал? Программировать в принципе можно и удаленно. "В принципе", потому что большинство нормальных фирм крайне настороженно воспринимают такой способ сотрудничества, предпочитая интернет-технологиям старый добрый офис и постоянный штат. К тому же оплата удаленных программистов в несколько раз ниже, зато выбор работы у них шире. Трудоустраиваются даже те, кто знает Фокс-домостос или древний Кобол, не говоря уже о системном программировании, машинных кодах и ассемблере. К тому же одновременно можно устроиться в трех-четырех местах. Совокупная зарплата практически не ограничена. Главное - обладать работоспособностью!

Арабский вариант добывания денег

Fight Club - отнюдь не только фильм. Почитай книгу Паланика

Тысячи специалистов, работавшие над "Боингом", были выброшены после того как птичка начала летать.

Чисто технически можно присоединиться к любому готовому проекту. На форумах рассказывают кучи историй о том, как молодой программист нашел в известном компиляторе баг и затем стал бета-тестером на постоянной основе. Или о том, как журналист написал документацию к продукту, устроившись техническим писателем. Судя по всему, это легенды, выдуманные для самоуспокоения. С бета-тестерами чаще всего расплачиваются новыми бета-версиями, а что же до технических писателей... а ты попробуй написать документацию удаленно! Без программиста под рукой! Есть менюшка, но что она делает - непонятно. А половина возможностей вообще скрыта в "недокументированных" сочетаниях горячих клавиш. В недокументированных, потому что программист просто забыл включить их в документацию, но сам он (и все его коллеги) пользуется ими постоянно...

Как вариант, можно заняться внедрением чужих продуктов у себя на местах. Но здесь больше бизнеса, чем самого программирования (ну разве что выпросить исходные тексты с подпиской о нераспространении с целью доработки и адаптации под конкретные требования "аборигенов", да только кто их даст?!) Но для этого нужен солидный начальный капитал. Уж лучше устанавливать пользователям Windows/*nix, благо при нынешней компьютерной "грамотности" спрос на таких "программистов" необычайно велик и со временем будет только расти. Но к обсуждаемой нами теме это не имеет никакого отношения.

Рынок "вольных хлебов", на котором пасутся свободные копейщики, широк и могуч. Он простилается от Аляски до глухих сибирских пенат. В отличие от сотрудничества на контрактной основе, предполагающего более или менее длительные отношения, свободный копейщик ориентирован на краткосрочный заказ. Защитить продукт от копирования. Перенести программу с Perl на C и т.д. Услуги свободных копейщиков резко полярны: или высокоинтеплектуальная задача (как, например, в случае с защитой), с которой штатные сотрудники компании-заказчика справиться не в состоянии, или рутинная работа, которую дешевле перебросить на "пионеров", чем напрягаться самим (как, например, в случае с переносом). Тем не менее пестрое племя свободных копейщиков в основном состоит из профессиональных программистов, продающих свои знания по цене кокаина. Остальные в этом мире просто не выживают, уходя в "серьезные" фирмы на постоянную работу. Любой аналитик подтвердит, что 90% дохода фирма получает от 10% специалистов (а если брать таких гигантов, как Intel или Microsoft, то соотношение и вовсе окажется 99:1). Но ведь этим 90% тоже нужно что-то платить! И это "что-то" приходится отрывать от специалистов.

Существует два диаметрально противоположных способа оплаты: единовременная выплата и отчисление определенного процента с продаж программного пакета (аппаратно-программного комплекса), в создании которого ты принимал участие. Нанимателя охотнее идут на единовременную оплату, размер которой в зависимости от специфики заказа колеблется от сотен до десятков тысяч долларов. Величина отчислений (так называемых royalty) редко превышает 10% от розничной/оптовой стоимости одного экземпляра ПО, но даже 1-3% лучше, чем совсем ничего. Тут все зависит от раскрутки продукта и объемов продаж. На отчислениях можно неплохо заработать, а можно и потерять (попробуй проконтролировать, сколько копий продано - одна или миллион!), в то время как сумма единовременной выплаты гарантирована. В общем, royalty - это журавль в небе, а единовременная выплата - синица в руке. Мне журавли нравятся больше, хотя здесь не обходится без разочарований и обманов.

Вместо того чтобы работать на "дядю", некоторые программисты предпочитают трудиться на самих себя. Они в одиночку (или тесным коллективом) создают утилиты или даже целые программные комплексы, приторговывая ими через интернет. Считается, что это самый прибыльный и наиболее перспективный путь. Не тут-то было. Просто так сесть и настрочить ша- »

Оригинал статьи (нерезаную и нецензурированную версию) ты сможешь найти на диске, прилагающемся к журналу.

Будь жадными до информации, как бурундук до желудей. Информация - штука такая, никогда нельзя сказать наперед, понадобиться тебе она или нет.

Программистская валюта ;)

Искать free lance на job.ru - не самая лучшая затея

ровары не получится. Тут идеи нужны! А у кого они есть? Рынок программного обеспечения забит всевозможными утилитами до отказа. Все, что только можно было придумать, уже придумано и реализовано в десятках конкурирующих меж собой софтин. Корпорации тоже не сидят без дела, постепенно прибирая новые ниши рынка к рукам. Еще недавно IE не умел сохранять страницы на диск целиком и не поддерживал докачку, что давало пищу шароварным программистам, но теперь... Старое поколение ставит ReGet скорее по инерции, чем по необходимости (V.92, DSL делают докачку все менее и менее актуальной), а web-страницу быстрее найти в интернете, чем на своем замусоренном жестком диске.

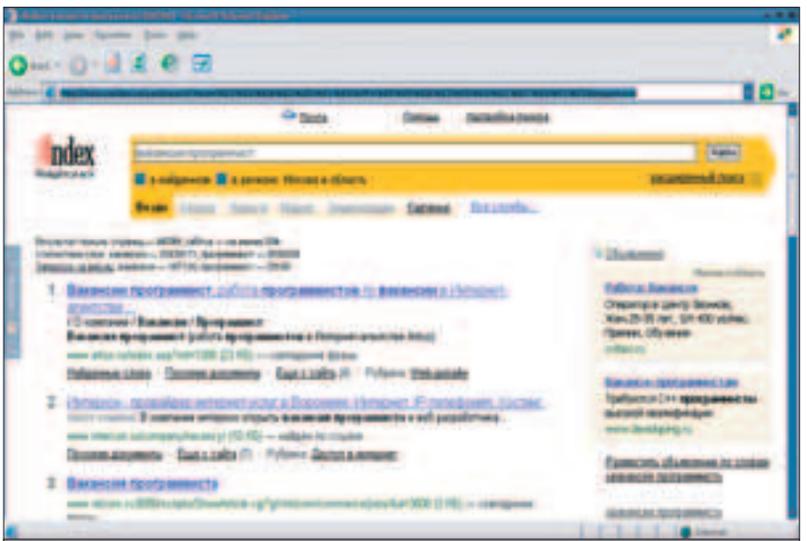
К тому же работающая программа еще не есть готовый продукт, предполагающий рекламу, маркетинг, внедрение, техническую поддержку, обучение пользователей и т.д. Без этого об объемах продаж можно забыть. Доход большинства шароварщиков невелик. Удачных проектов единицы, и нечего на них ориентироваться (ты бы еще Билла Гейтса в пример привел ;)). Но дело даже не в этом. Разработка шароваr - широкая предметная область, охватывающая все спектры программистской деятельности: от дизайна до кодинга. Узким специалистам здесь не место, а широкими специалистами за два дня не становятся. Я не беру в расчет тех молодых людей, что пишут в своем резюме невообразимо длинный список языков программирования и программных пакетов, в которых они как бы умеют работать. Без разделения и специализации труда создать конкурентоспособный продукт вряд ли получится, для этого нужна фирма, пусть маленькая и мобильная, но все-таки фирма, а не пара небритых мужиков с горящими глазами и залитой пивом кейбордой. Еще нужны большие вложения (надо же на что-то жить, пока создается продукт) без гарантии их окупаемости. Мрачноватая картинка получается? Индийский вариант в России не удался. Валового продукта не появилось. Так, отдельные яркие утилиты и приложения: RAR, FAR, Fine Reader. Зато мы создаем лучшие в мире компиляторы (филиал Intel в Нижнем Новгороде), космические симуляторы (филиал Maris Multimedia в подмосковном Королеве), системы распознавания рукописного текста, нашедшие применение в Palm. Список можно продолжать бесконечно. Куда ни плюнь, всюду мы. Так что крест на себе ставить не нужно.

Конечному пользователю по большому счету абсолютно все равно, за счет каких именно физических механизмов обеспечивается должный уровень функциональности окружающих его устройств. Все равно он видит один лишь интерфейс.

Учебники в стиле "Язык Си++ для начинающих" приносят намного больше вреда, чем пользы, и зачастую необратимо калечат мышление новичков.

ЧЕМ ЗАНЯТЬСЯ?

■ До недавнего времени наиболее доходной областью было проектирование и программирование баз данных. Сначала это были простенькие приложения, затем они объединились



Поиск выдал неплохой список компаний с вакансиями программиста

в локальные сети, а сети - в геоинформационные системы. В России с ее вечно меняющейся законодательной базой финансоотчетные программы превратились в настоящую кормушку, питающую полчища разработчиков, лабающих кривой код на потоке. Теперь это в прошлом. Бухгалтеры пересели на 1С, и чистые программисты оказались невостребованными.

Зато сейчас интенсивно развивается телефония. Приложения для сотовых телефонов расходятся на ура (даже если это всего лишь "Питон" или "Тетрис" в стиле начала 80-х). Учишь Java, скачиваешь SDK с сайта производителя - и вперед! Создается множество мини-АТС на базе обычных персоналок со звуковым кодеком внутри. Берешь готовый кодек, написанный, как правило, на Си++, оптимизируешь его (чтобы втиснуть в быстродействие ЦП как можно больше каналов), клепаешь аппаратную часть и... железяка готова!

А вот сетевое программирование приходит в упадок. Работа утекает в крупные конторы, повсеместно внедряются готовые решения, армия свободных копейщиков ожесточенно дерется в конкуренции, и доходы, соответственно, падают. Технологии программирования сменяются одна за другой, приобретенные навыки быстро устаревают. Последующие несколько лет, судя по всему, пройдут под знаменем .NET'а (которому пророчат целую эпоху), правда, никаких гарантий на этот счет нет, так что вопрос о выборе средств разработки остается открытым.

Дырявость сетевого обеспечение и его неумелое использование притягивает как хакеров, так и тех, кто с ними борется. В высших учебных заведениях уже появилась такая специальность, как "Безопасность". Чему там учат - непонятно. Хакер - это вам не прыщавый подросток! Это высококвалифицированный специалист с огромным опытом, развитой интуицией, нетривиальным мышлением, просчитывающий поведение машины на несколько шагов вперед. Роберт Моррис (известным своим червем) до самого известного из своих занятий занимался тем, что переписывал системы безопасности для больших компьютеров, и с дырами он был знаком не понаслышке. Предотвратить нападение червя мог только другой Моррис, а не "эксперт по безопасности", прослушивавший курс лекций, но никогда не заглядывавший в исходный код sendmail'а. Чтобы защититься от хакеров, необходимо иметь выдающуюся эрудицию и просчитывать все на десять-двадцать шагов вперед. Все, что может сделать "эксперт" - выявить грубые ошибки конфигурации системы (дырявый сервер, неправильно сконфигурированный брандмауэр). Заниматься консалтингом прибыльно, но рискованно. Иногда клиент может и по морде дать во всей физической прямоте этого слова. Ну да! С него взяли деньги, а через некоторое время атаковали! Брать плату за конкретно обнаженные дыры намного проще. Особых познаний и навыков здесь не требуется. Можно, например, дизассемблировать приложения на предмет поиска ошибок переполнения, просматривать исходные тексты, проверять скрипты или специализироваться на TCP/IP-протоколах: подделке обратных адресов, перехвате трафика, обходе брандмауэров и т.д. Это увлекательно и высокооплачиваемо (в среднем по $500 за дыру, если предложат меньше - ну их).

Защита программного обеспечения, похоже, переживает свои последние дни. Программные решения уступают место аппаратным, а демократия сменяется жестким тоталитарным режимом, расценивающим пиратство как терроризм, а терроризм - это пожизненное заключение или расстрел. Тем не менее хорошие специалисты без работы не останутся и всегда найдут себе применение. Взять те же лазерные диски. Обычная такса за защиту: ~5% от стоимости тиража. Разумеет-

ся, приведенная цифра очень условна. Тут все зависит от уровня защиты, стоимости дисков и размеров самого тиража. За массовый продукт можно взять и 1% - не прогадаешь. А 300-500 экземпляров можно и не защищать: не окупится, разве что поставить типовую защиту, но тогда ее сразу взломают.

Вовсю идет дизассемблирование ПЗУ и программных модулей для похищения оригинальных алгоритмов, восстановления структуры файлов данных или протокола обмена. Эта работа неплохо оплачивается, хотя она и не совсем законна. Наше законодательство не запрещает дизассемблирование, а его результатами все равно будешь пользоваться не ты, так что вся ответственность ложится на нанимателя. Правда, после этого в Америку можно въехать только чучелом или тушкой. Есть риск, что арестуют прямо в аэропорте. Уж лучше отправляться в Южную Корею. Там наших любят. Не в смысле чтобы "кушать" (это делают в Аффрике), а как специалистов по разработке 3D-шутеров и вообще.

Программирование игр - весьма увлекательное и к тому же неплохо оплачиваемое занятие. Игры, естественно, должны быть трехмерными и фотореалистичными, а другие никому не нужны. К тому же быстродействие современных процессоров не бесконечно, и эффективность исполнения кода вырывается на первый план. Опять-таки, необходимо хорошо знать физику и математику. Без этого можно запрограммировать разве что крестики-нолики. А отсюда недалеко до разработки систем моделирования (движение звезд в галактике, развитие атмосферных циклонов и фронтов, развитие напряжений внутри сварочного шва). Заказчиками выступают либо институты, либо коммерческие корпорации. Кстати говоря, моделирование - это как раз та сфера, в которой Россия всегда была традиционно сильна. У индусов кишка тонка.

Восстановление данных - еще одна перспективная область, гарантирующая, что без куска хлеба специалист не останется. Даже в масштабах небольшого уездного городка проблемы с жесткими дисками и оптическими носителями случаются регулярно. Конечно, для восстановления данных на физическом уровне необходимо весьма дорогостоящее оборудование, но в подавляющем большинстве случаев разрушения носят логический характер и для их восстановления достаточно иметь редактор диска плюс пару-тройку утилит собственного написания. Автоматизированные докторы типа Easy Recovery - это все фигня. Для домашнего использования вполне сойдет, но брать за такое "восстановление" деньги...

Наблюдается рост спроса и на программирование микроконтроллеров, в которых доминирует ассемблер, C, форт и машинные коды. Работа приносит такое удовольствие, что брать за нее деньги становится просто стыдно. Но ведь дают! Правда непостоянно. Иной раз за месяц не поступает ни одного заказа, тогда на пропитание приходится зарабатывать сборкой домашних кинотеатров (они сейчас популярны в народе): просто берем slim-корпус, отрываем от него мышь, монитор и клаву, пишем простенький загрузчик Linux'a, автоматический распознаватель формата диска и подцепляем к плееру пульт управления по ИК. Да много чего сконструировать можно - была бы фантазия! Возможностей для самореализации - море. Выбирай на вкус. Слухи о безработице и невостребованности программистов сильно преувеличены. Отмирают одни специальности, но на их месте расцветают другие, подтвер-ждая естественный круговорот. Так что работу себе не найдет только гурман или ленивый.

Что же до администрирования, сборки и ремонта компьютеров - это перспективные и притом бурно развивающиеся области, но к программированию они не относятся, а потому в моем обзоре не затрагиваются. К слову сказать, администратор - это в первую очередь хороший хозяйственник и только потом знаток сетевых протоколов и оси. Гуру, настраивающие систему так, чтобы она работала и не падала, вынуждены конкурировать с большой армией начинающих "администраторов", даже не дотягивающих до звания продвинутых пользователей. При всех своих недостатках Windows ставится с полпинка и даже как бы работает. Необходимость в администрировании осознается начальством лишь тогда, когда Windows его конкретно поимеет, а это случается не со всеми и не всегда... То же самое со сборкой. Выткнуть материку в корпус - большого ума не надо. А грамотно сбалансированная, дешевая, безглючная и высокопроизводительная конфигурация никому не нужна. То есть нужна, конечно, но переплачивать за это никто не станет. Ремонт электроники экономически невыгоден. Поломанный блок легче выкинуть и заменить новым, более современным, чем горбатиться над паяльником. "Обслуживание" компьютеров, сводящееся к переустановке Windows, вытесняет качественный сервис, когда упавшая система поднимается без радикальных перемен. Да что там говорить... Специалисты нужны лишь на гребне волны. На острие прогресса. На передовой линии фронта. В тылу им делать нечего. В тылу обитают MS Word и сексапильные секретарши.

FREE LANCE VS STEADY JOB

■ Стоит ли искать постоянную работу или лучше оставаться на вольных »

Место, где зарабатывают шароварщики

Администратор - это в первую очередь хороший хозяйственник и только потом знаток сетевых протоколов и оси.

хлебах? Все зависит от психотипа личности. Кто-то предпочитает кочевую жизнь, а кто-то оседлую. Не будем делить программистов на "правых" и "виноватых", а лучше расскажем о каждой из сторон поподробнее, попутно отмечая некоторые не вполне очевидные проблемы, с которыми придется столкнуться в пути.

Свободных копейщиков (они же free lancer'ы) можно встретить в любой точке мира - от столичной квартиры до глухой провинциальной норы. Интернет стирает границы и уравновешивает город с деревней в правах. Даже находясь в шалаше, можно скачивать стандарты и спецификации, общаться с коллегами по всему миру, оставаясь при этом запертым в четырех стенах, которые что там, что тут одинаковы. Свободный график (точнее, полное отсутствие такового), богатый ассортимент начальства, никакой карьерной лестницы со свойственными ей пороками и извращениями. Понятие "надо" кажется абстрактным и бесконечно далеким, уступая место "мне нравиться" и "вот это кайф". Для творческий людей с широким спектром скачкообразно меняющихся интересов лучших условий работы нельзя и придумать.

Теперь перейдем к недостаткам. Чтобы удержаться на плаву, свободный копейщик должен быть подвижным, как ртуть. Заказов много - только успевай, но на блюдечке с голубой каемочкой их никто не принесет (во всяком случае на первых порах). Значит, нужны обширные связи. Тематика заказов самая разнообразная: от микроконтроллеров до телефонии, и узким специалистам тут приходится туго. К тому же в последнее время наметилась неприятная тенденция оттока заказов в корпорации и в крупные программистские фирмы, у которых свой укомплектованный штат.

Вместо райских садов нас встречают дикие джунгли, живущие по принципу "волка ноги кормят". Обычно заказы ходят косяками: то вообще никакой халтуры нет, а то кааак сыпанет! Нахватаешь ее на радостях (после месяца ничегонеделанья любая работа встречается с трудноскрываемым энтузиазмом), а потом чешешь репу и думаешь: "Когда же я все это делать буду?" Какой там сон! Хорошо если вздремнешь на клавиатуре часок-полтора. Какая еда! Жуешь бутерброд вместе с промасленной распечаткой, попутно обдумывая архитектуру будущего проекта и стуча по клаве свободной рукой! Это на постоянной работе можно прийти, лениво почитать журнальчик, пофлиртовать с секретаршей, а потом, закрывшись у себя в кабинете, основательно поDOOMать или почаститься до конца трудового дня.

У свободных копейщиков расклад совершенно иной. Времени на личную жизнь практически ни у кого не хватает, что часто приводит к жестоким размолвкам в семье. Ты пробовал программировать при жене? Я пробовал. Вынеси то, подай это, сделай все наоборот. Ты меня совсем не любишь и т.д. В общем, развелся. Не могу удержаться, чтобы не привести еще одну цитату: "...молодой инженер уходил на работу к восьми утра, работал без перерыва на обед, уходил с завода часов в семь, приезжал домой, полчаса играл с ребенком, ужинал с женой, ложился в постель, быстро занимался с ней любовью, затем вставал и, оставив ее в темноте, уходил на два-три часа за свой стол, чтобы поработать над парой вещей, которые взял с собой. Он мог, уходя с завода, заглянуть в Wagon Wheel и выпить пива... Вернувшись домой в девять, когда ребенок уже уснул, ужин холодный, а жена еще холоднее, он пытался объяснить ей что-то, а в голове у него вертелись совсем другие мысли: LSI, VLSI, альфа-поток, прямое смещение, паразитические сигналы... К тому же в Wagon Wheel он встретил сексапильную пышку из Signetics, и она его прекрасно понимает," - Тим Джексон, "INTEL - взгляд изнутри".

Эту книгу читать необходимо! Желание связываться с крупными корпорациями сразу же пропадет. Сильнее всех страдают русские программисты, не имеющие никакого иммунитета против чумы западного мира. Если говорить кратко, там берут специалиста, прогоняют через соковыжималку и выбрасывают, как ненужный хлам на свалку. Если же он одумается и попытается свалить из компании в собственный бизнес - его разоряют. Мрак. Полный.

ЗАКЛЮЧЕНИЕ

Так все-таки можно зарабатывать на программировании или нет? Да как тебе сказать. В глубинке молодой специалист, получающий порядка $500 в месяц и не обремененный подхалимажем перед вечно недовольным начальством, вызывает у окружающих смесь зависти с восхищением. Но и работать ему приходится ой-ой-ой! В столице и промышленных центрах страны эта цифра вызывает снисходительную улыбку с одобряющим похлопыванием по плечу: "...может, тебе одолжить, а? У тебя вид какой-то запущенный".

Профессия программиста уже утратила свой мистический ореол, и ее популярность тает прямо на глазах. Случайных людей здесь становится все меньше и меньше. В программирование идут преимущественно те, кому интересно проектировать структуры данных, листать потрепанную документацию, ковырятся в отладчике... Чем качественнее код, тем ниже его доходность (как это ни прискорбно, но факт!), однако погоня за личным обогащением опускает вычислительную технику в глубокую яму. Чтобы многолетние отложения глюкавого кода не рухнули окончательно, нужно забыть о деньгах и вспомнить, что Россия - это страна с традиционно высокой инженерной культурой и неординарными людьми.

НЕ ХВАТАЕТ ЧЕГО-ТО ОСОБЕННОГО?

Играй просто!
GamePost

Star Wars Galaxies: An Empire Divided - Collector's Edition	EverQuest II Collector's Edition	Lineage II Collector's DVD Edition
$99.99	**$155.99**	**$69.99**

*

WarCraft Action Figure:
Grom HellScream **$42,99**

У НАС ПОЛНО ЭКСКЛЮЗИВА

* Эксклюзивные игры
* Коллекции фигурок из игр
* Коллекционные наборы

Xbox
$239.99

Тел.: (095) 928-0360
(095) 928-6089
(095) 928-3574
www.gamepost.ru

GAMEPOST

Алексей Башкеев (botan@dezcom.mephi.ru)

В ИНОСТРАННУЮ ФИРМУ ТРЕБУЮТСЯ...

КТО ТАКИЕ ИНОСТРАННЫЕ РАБОТОДАТЕЛИ И ЧТО ОНИ ДЕЛАЮТ НА НАШЕМ РЫНКЕ ТРУДА?

В объявлениях о работе довольно часто встречаются магические слова "иностранная компания". Стоит ли с ними связываться и какая материальная выгода в этом для программиста?

МИРОВОЙ РЫНОК ТРУДА

Возможно, это звучит сложно и заумно, но на самом деле все просто. Ты, я надеюсь, слышал, что в разных странах одна и та же работа стоит разных денег. В некоторых странах Африки $50 в месяц - хорошая зарплата, а у нас за такие деньги вряд ли захочется работать. В Европе и штатах - тем более. А чего хотят работодатели? Правильно - они хотят получить качественный программный продукт за минимальные деньги. Следовательно, для них очень выгодно открыть филиал где-нибудь в Зимбабве или Никарагуа и платить там "очень хорошие зарплаты". Но к сожалению, программисты в Зимбабве и Никарагуа не очень выдающиеся.

Вот эти "иностранные конторы" приходят и к нам, но не только к нам. Еще в Индию, Белоруссию, на Украину... В этих странах программисты стоят в несколько раз дешевле, чем "их зарубежные аналоги". Этот процесс "минимизации издержек" и приводит к "глобализации экономики". Естественно, при этом не удел остаются программисты западных стран. Вот они и ходят с демонстрациями вместе с работниками других отраслей производства. Но это, так сказать, их трудности.

А где наше место в этой "пищевой цепочке"? Уровень заработной платы на Украине и в Белоруссии существенно ниже, чем, к примеру, в Москве. Но там "рынок труда" тоже меньше, не так много хороших специалистов. В частности, поэтому Москва и Питер хоть и стоят дороже наших коллег из ближнего зарубежья, спрос на нас существует. В общем, факт остается фактом: работодатели приходят к нам и хотят, чтобы мы для них что-то делали.

В ЧЕМ ВЫГОДЫ РАБОТЫ "НА НИХ"?

ЗАРПЛАТА

Естественно, "чуть" выше плюс еще одна особенность. Ты, наверное, слышал по телевизору про 13%. Это же телевизор тактично молчит о том, что кроме тех 13%, которые платишь ты, твой работодатель платит еще много разных налогов. Другими словами, чтобы платить тебе, например, $1000 "белой" зарплаты, работодателю надо платить еще $400-500 налогов. Поэтому наши работодатели платят "черную зарплату" - экономят на налогах. Не вдаваясь в налоговую схемотехнику, иностранная компания может легально платить тебе $1000 белой зарплаты, отдавая государству только $130 (те самые 13%). В общем, обоюдная выгода.

КОЛЛЕКТИВ И АТМОСФЕРА

Во многих иностранных фирмах очень жесткие условия труда и атмосфера, которая не по душе нашему брату: все делается для того, чтобы поднять эффективность производства. Ни одного лишнего человека. Работа жестко поделена и сроки жестко выставлены. Строгая дисциплина, системы штрафов, бонусов... В общем - никакой халявы. Страшновато звучит? На самом деле у этой медали тоже две стороны. Весь этот ужас - следствие того, что менеджмент в таких фирмах, как правило, лучше, чем в наших. Получить максимально качественный продукт в максимально сжатые сроки, потратив при этом минимум средств. И те "порядки", которые у нас только начинают зарождаться, в западных конторах уже получили хорошее развитие.

В общем, описывать "тамошнюю" атмосферу можно долго, но лучше один раз попробовать, чем сто раз прочитать. Многие из тех, кто работал там, говорят, что "поварившись" в такой каше, получаешь ценный опыт. К тому же соответствующая строчка в резюме позволяет рассчитывать на более высокий уровень оплаты труда и в отечественных конторах.

А НЕ СЪЕЗДИТЬ ЛИ ТУДА?

Про то, чем отличается туризм от эмиграции, и про русские березки много писать не буду. При всех недостатках заграничной жизни там очень можно жить. Например, из 20000 разработчиков кампуса Microsoft в Редмонде минимум 1000 говорят на русском языке. Так что процесс мировой русификации идет полным ходом. Кстати, есть еще один момент, который надо учитывать. Важен не только уровень зарплат, но и "прожиточный минимум". Безусловно, если задаться целью сэкономить... Можно отказаться от страховки, выбрать квартиру похуже и т.д. При желании такую информацию можно получить пообщавшись с коллегами, работающими там (думаю, сможешь нарыть их координаты, если очень захочешь).

Некоторые компанию играют в открытую и на своих сайтах сразу публикуют и уровень заработной платы, и калькулятор "уровня жизни", для того чтобы соискатели сами могли оценить, "что по чем". Ходят слухи, что к себе "они" приглашают только лучших. Так что если ты работал на них на их территории... В общем, ты станешь еще более востребованным специалистом.

ХОЧУ ПОРАБОТАТЬ НА "НИХ", А НАДО ЛИ МНЕ...?

Во-первых, успокойся и глубоко вдохни. Для того чтобы обеспечить тебе все условия для полноценного творчества, работодатель возьмет на себя большую часть проблем. Они это умеют. Какую карточку надо заводить, какие документы надо собрать, как получить визу, если таковая понадобится - тебе все расскажут. Такое "разделение труда" имеет много дос-

Типичное объявление о работе

тоинств, и благодаря ему ты занимаешься только своим делом - программированием.

ЧТО-ТО ВСЕ СЛИШКОМ РАДУЖНО...

На самом деле, конечно, нет. Во-первых, работать придется по-настоящему много. Во-вторых, это все-таки чужая контора, которая находится на нашей территории только потому что это выгодно. Изменятся условия - работодатель уйдет туда, где лучше. А вот наш работодатель всерьез и надолго на нашем рынке. Тут проверенные кадры могут на многое рассчитывать. Беспроцентные кредиты в счет зарплаты, поручительство в банке, социальные льготы (договоры с домами отдыха, детсады, вопросы с недвижимостью...) - много разных "бонусов", надо заметить.

В нас верят!

Чем больше компания - тем больше список. Крупные компании (имеющие крупные отделения) предоставляют массу льгот, недоступных небольшим компаниям. Они "берут оптом", поэтому медицинское обслуживание, корпоративный спортзал, корпоративная столовая и т.д. им обходится дешевле в расчете на одного человека. Крупные иностранные конторы не имеют больших представительств у нас, за редким исключением.


Некоторые компании играют в открытую

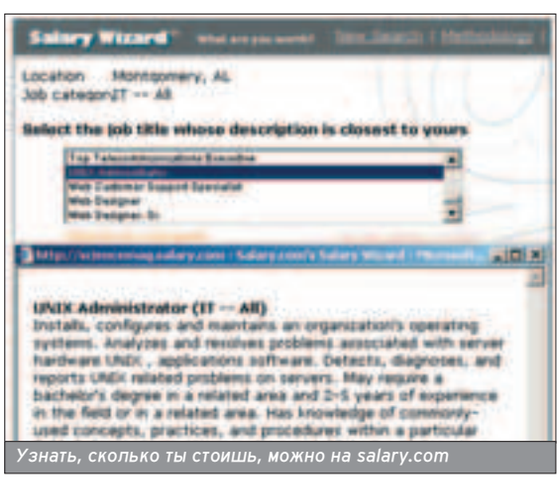
А сейчас читай то, что вселит в тебя порцию оптимизма. В интернете есть такой сайт www.longbets.org, на котором прогнозируют развитие разных сфер деятельности (наука, техника, политика и др.) и делают соответствующие ставки. Под номером "5" там есть очень интересный пункт (www.longbets.org/5). Esther Dyson (если кому это имя ничего не говорит - в Яндекс! Таких людей знать надо) поставила $10000 на то, что к 2012 году Россия станет державой №1 по производству программного обеспечения. Будем надеяться, что мы оправдаем ее ожидания. Вот только с ней согласна всего лишь пятая часть проголосовавших... (срочно регистрируемся на этом сайте и меняем это соотношение кардинальным образом!).

А СКОЛЬКО Я СТОЮ ЗА ГРАНИЦЕЙ?

Приведу несколько ценных ссылок для людей, интересующихся такими вопросами. Стоит иметь в виду, что зарплату у них принято мерить не за месяц, а за год.

www.ticker.computerjobs.com - масса заполненных анкет и ценная статистика.

http://recruit.sciencemag.org/feature/salarywizard.shl - очень полезная ссылка, показывает уровень дохода в США (отсортировано по штатам) инженеров, программистов и ученых. Себя я там нашел, надеюсь, ты тоже найдешь что-нибудь полезное.



Узнать, сколько ты стоишь, можно на salary.com

МНЕНИЕ ЭКСПЕРТА

Евгений "Firstborn" Рогов, аналитик и специалист по качеству ПО

Многие задумываются о работе за рубежом - почему? Как правило, из-за сугубо финансовых соображений, то есть из-за вполне понятного и в общем-то приветствуемого желания улучшить свое благосостояние как можно более быстрыми темпами. Нередко такое решение проблемы может оказаться вполне логичным: общеизвестно, что экономическая ситуация в разных странах (и даже в разных городах одной и той же страны) может очень сильно различаться. Один и тот же специалист в разных точках планеты сможет претендовать на зарплаты, различающиеся на порядок, а то и больше. Не спорю, что человек, зарабатывающий $300 в месяц и узнавший о $3000, которые получает обладающий такой же квалификацией житель другой страны или другого города, серьезно задумается о переезде. Все это распространяется не только на IT.

Однако в области информационных технологий есть своя специфика, потому что продукт труда (например, написанный программистом исходный код) может быть без труда передан на любое расстояние практически без дополнительных затрат! В результате физический переезд (временный или нет) в другую страну или в другой город уже не будет столь очевидным решением, тем более что он сопряжен с серьезными бытовыми хлопотами. Не стоит забывать и о том, что там, где наш специалист станет зарабатывать $3000 вместо $300, он вряд ли будет тратить столько же, сколько и раньше: на новом месте возрастет и стоимость жизни. Так что, судя по всему, одним из наиболее удачных вариантов для профессионала в области информационных технологий был бы, так сказать, виртуальный переезд в такое место, где его знания и умения оплачиваются лучше. Таким образом, физически находясь в своем родном городе, не отрывая себя от родных и друзей, можно значительно увеличить свои доходы за счет удаленной работы, пусть даже она не является основной.

ЗАКАЗ ЖУРНАЛА В РЕДАКЦИИ

Бесплатный телефон по всем вопросам подписки
8-800-200-3-999
(включая абонентов МТС, БиЛайн, Мегафон)

ВЫГОДА
Цена подписки на 20% 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 с курьером

Стоимость заказа на «Хакер Спец» + CD

115р	за номер (экономия 40 рублей*)
690р	за 6 месяцев (экономия 240 рублей*)
1242р	за 12 месяцев (экономия **620** рублей*)

Стоимость заказа на комплект «Хакер Спец»+CD + «Железо»+CD

189р	комплект на 1 месяц (экономия 85 рублей*)
1071р	комплект на 6 месяцев (экономия 510 рублей*)
2016р	комплект на 12 месяцев (экономия **1250** рублей*)

* экономия от средней розничной цены по Москве

ЗАКАЖИ ЖУРНАЛ В РЕДАКЦИИ И СЭКОНОМЬ ДЕНЬГИ

подписной купон

Прошу оформить подписку:

☐ на журнал Хакер Спец + CD

☐ на комплект Хакер Спец+CD и Железо+CD

на ☐ месяцев

начиная с _____ 2005 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 рожд. ☐☐. ☐☐. ☐☐ г.
день месяц год

АДРЕС ДОСТАВКИ:

индекс

область/край

город

улица

дом корпус

квартира/офис

телефон ( )
код

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 Международный Московский Банк, г. Москва

р/с № 40702810700010298407

к/с № 30101810300000000545

БИК 044525545 КПП - 772901001

Плательщик

Адрес (с индексом)

Назначение платежа	Сумма
Оплата за «_____» с _____ 2005 г. (месяц)	

Ф.И.О.

Подпись плательщика

Кассир

Квитанция

ИНН 7729410015 ООО «Гейм Лэнд»

ЗАО Международный Московский Банк, г. Москва

р/с № 40702810700010298407

к/с № 30101810300000000545

БИК 044525545 КПП - 772901001

Плательщик

Адрес (с индексом)

Назначение платежа	Сумма
Оплата за «_____» с _____ 2005 г. (месяц)	

Ф.И.О.

Подпись плательщика

Кассир

Как оформить заказ

1. Заполнить купон и квитанцию
2. Перечислить стоимость подписки через Сбербанк
3. Обязательно прислать в редакцию копию оплаченной квитанции с четко заполненным купоном любым из перечисленных способов:

- по электронной почте: subscribe@glc.ru;
- по факсу: 924-96-94;
- по адресу: 107031, Москва, Дмитровский переулок, д. 4, строение 2, 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 можено оформить с ноября.

По всем вопросам по подписке звони бесплатно по телефону 8-800-200-3-999 (в том числе с мобильных телефонов сетей МТС, БиЛайн, Мегафон).
Вопросы по подписке можно задавать по e-mail: info@glc.ru

Подписка для юридических лиц

Москва: ООО «Интер-Почта», тел.: 500-00-60, e-mail: inter-post@sovintel.ru
Регионы: ООО «Корпоративная почта», тел.: 953-92-02, e-mail: kpp@sovintel.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.
www.interpochta.ru

Content:

Андрей Каролик (andrusha@real.xakep.ru)

ОБЗОР КНИГ

ЧТО ПОЛИСТАТЬ

В Сети можно откопать много информации, но далеко не любую. Отличным доказательством тому служит недоступность в интернете множества невероятно интересных книг. Тем не менее пренебрегать информацией только из-за того что ее нет в электронном виде по меньшей мере глупо.

УСПЕШНЫЙ IT-АУТСОРСИНГ

М.: КУДИЦ-ОБРАЗ
2004
Элизабет Спарроу
288 страниц
Разумная цена: 153 рубля

Книга будет интересна как тем, кто работает с ресурсами на стороне, так и тем, кто предлагает эти ресурсы. Если проект достаточно сложный, а денег приносит немало, стоит подумать над тем, чтобы не экономить на качестве работы и отдать часть проблем на решение сторонним программистам. Но этот шаг таит в себе множество проблем: от плохого понимания задачи и затягивания сроков исполнения до полного провала всего проекта. В книге рассматриваются потенциальные проблемы аутсорсинга, их зависимость от моделей аутсорсинга, построение эффективных отношений с поставщиком услуг аутсорсинга. Большая часть примеров предназначена для тех, кто ищет исполнителей на стороне. Правильно построишь процесс - и можно спокойно загорать у моря.

ЭЛЕКТРОННЫЙ БИЗНЕС

М.: ДМК Пресс
2003
Смирнов С.Н.
240 страниц
Разумная цена: 128 рублей

Бизнес все активнее обращается к сфере электронной торговли продуктами и услугами. Интернет-технологии позволяют обеспечить взаимодействие с максимально широкой и платежеспособной аудиторией. Но для использования всех этих благ нужно понимать основные модели электронного бизнеса, используемые платежные системы и способы решения проблем, которые обязательно возникнут при поиске необходимой информации. Обо всем этом ты сможешь прочитать в этой книге. Как бонус - закон "Об электронной цифровой подписи".

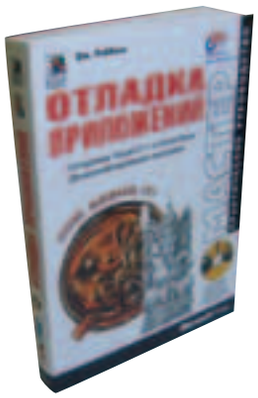
MICROSOFT .NET В ПОДЛИННИКЕ

СПб.: БХВ-Петербург
2004
Дубовцев А.В.
704 страницы
Разумная цена: 254 рубля

О платформе .NET и технологии Microsot .NET Framework не слышал только глухой. Все больше программистов переходят на эту платформу, привлеченные ее многочисленными возможностями. Эта мухобойка подробно описывает архитектуру и внутреннее устройство .NET, содержит массу примеров программирования на популярных языках C#, VB.NET, MC++ и IL. Для быстрого усвоения материала примеры простые и понятные. Отдельно рассмотрены недокументированные возможности .NET, которые могут вполне пригодиться на практике.

ОТЛАДКА ПРИЛОЖЕНИЙ

Тестирование и отладка приложений - очень важный процесс, который сопровождает раз-



СПб.: БХВ-Петербург
2001
Роббинс Дж.
512 страниц
Разумная цена: 158 рублей

работку любой программы. В этой замечательной книге рассматриваются методы отладки, трассировка, блочное тестирование, прерывание и пошаговый проход, удаленная отладка и автоматизированное тестирование, актуальное для больших проектов. Много примеров по дизассемблированию программ, работе с отладчиками Visual C++ и Visual Basic, по мультимашинной и мультипроцессорной трассировке, по многопоточным блокировкам. На прилагаемом диске все исходные коды, отладочные утилиты и полнофункциональный GUI-отладчик.

SHAREWARE: ПРОФЕССИОНАЛЬНАЯ РАЗРАБОТКА И ПРОДВИЖЕНИЕ ПРОГРАММ

Shareware – это, как известно, далеко не только программирование. Это раскрутка, поддержка, красивая подарочная упаковка – словом, все, что может помочь проекту быть прибыльным. В этой книге ты найдешь рекомендации по разработке интерфейса и оформлению документации, рекламе и технической поддержке своего программного продукта. Речь идет, конечно, о создании программ на профессиональном уровне. Ты узнаешь, почему программе необходим собственный сайт, как выбрать хостинг и зарегистрировать доменное имя, как организовать прием платежей, построить эффективное ценообразование и продвигать свою программу в массы. Больше знают - больше купят. Если, конечно, твоя программа стоит того.

СПб.: БХВ-Петербург
2003
Жарков С.В.
320 страниц
Разумная цена: 92 рубля

ПОЗИЦИОНИРОВАНИЕ: БИТВА ЗА УЗНАВАЕМОСТЬ

СПб.: Питер
2004
Джек Траут
256 страниц
Разумная цена: 79 рублей

Если ты придумал нечто уникальное, главное - кричать громче и быстрее, чтобы потенциальные потребители услышали, а конкуренты не успели скопировать. Если же ты делаешь что-то уже имеющее аналоги, придется делать упор на позиционирование и узнаваемость. Как ты узнаешь из книги, ключ к успеху в конкурентной борьбе - не лучший товар, а лучшая идея, которая и заставляет потребителей выбирать тебя, а не других. В книге рассказывается о феноменальных успехах и грандиозных провалах в области рекламы за последние два десятка лет. На этих примерах проще всего понять, как нужно рекламировать свой продукт и как это делать категорически не рекомендуется.

УПРАВЛЕНИЕ КОМАНДОЙ: КАК ЗАСТАВИТЬ ДРУГИХ ДЕЛАТЬ ТО, ЧТО ВАМ НУЖНО

СПб.: Питер
2004
Льюис Джеймс
160 страниц
Разумная цена: 64 рубля

Собрать команду на порядок проще, чем заставить эту команду добиваться определенных поставленных целей. Особенно непобедимы в этом отношении русские из-за особого менталитета раздолбайства и пофигизма :). Руководителю нужна мотивация и лидерские качества, с помощью которых он заставит всех делать то, что нужно. Книга поможет осознать основные принципы мотивации членов команды, эффективного общения внутри команды и креативного управления командой. Сможешь реализовать на практике - свернешь любые горы, иначе будет обычная серая команда без потенциала и будущего.

КАК УВИДЕТЬ ДЕНЬГИ НА ЭКРАНЕ МОНИТОРА

СПб.: Питер
2004
под ред. Сафина В.И.
220 страниц
Разумная цена: 139 рублей

Для непосвященного биржевые торги могут показаться чем-то непостижимым, но только на первый взгляд. Ничего сложного в торгах нет, просто чтобы торговать на финансовых рынках и зарабатывать, нужны навыки и опыт. Опыт купить невозможно - только получить на практике. А вот навыки можно. В этом случае за какие-то 139 рублей ты приобретешь первоначальные знания дилинга и понимание таинственных графиков, используемых для анализа рынка. А практиковаться можно на многочисленных онлайн-биржах, которых становится все больше и на многих из которых можно поиграть с ненастоящими деньгами, но по-серьезному.

■ Любые из описанных книжек, которые тебя заинтересовали, можешь заказать (по разумным ценам) не отрывая пятой точки от дивана или стула в букинистическом интернет-магазине "OS-Книга" (**www.osbook.ru**). Книги для обзора мы брали именно там.

Леонид Кофман (kofman@vlink.ru)

FAQ

ВСЕ, ЧТО ТЫ ХОТЕЛ ЗНАТЬ О SHAREWARE, НО БОЯЛСЯ СПРОСИТЬ

Q: Сколько времени можно писать первую версию программы?

A: Вообще-то если тебе не жалко времени - то бесконечно долго. Оттягивание момента релиза первой версии - обычное дело для начинающего. Человек боится показать свое детище людям, опасаясь, что они не оценят его стараний. Это нормально, но все-таки надо себя преодолеть. Поэтому я рекомендую через два-три месяца упорной работы, если программа уже выполняет свои основные функции, смело ее выпускать. Только на этом этапе лучше оставить ее бесплатной с пометкой "Beta".

Q: Какой, в среднем, должен быть период выпуска новых версий?

A: Это значение сильно различается в зависимости от особенностей программы, причем чем сложнее программа, тем, конечно, больше периоды. В среднем же для shareware-проектов период выпуска новых версии – один-три месяца. Однако надо четко соблюдать составленный график релизов, чтобы не было такого, что два месяца подряд программа обновлялась, а потом за полгода не было ни одной версии. Также рекомендую присмотреться к ближайшим конкурентам и выяснить, какой у них период выпуска релизов.

Q: Придется ли мне платить налоги?

A: Тут ответ - категорическое "да". Ты должен четко понимать, что shareware - это предпринимательская деятельность, которая в случае неуплаты налогов превращается в незаконную предпринимательскую деятельность со всеми вытекающими отсюда последствиями. Конкретные проценты я называть не буду просто потому что слишком много факторов, влияющих на их размеры. Скажу только, что надо сразу решить, будешь ты выступать как физическое лицо или как юридическое. У каждого варианта есть свои плюсы и минусы, но налогов, конечно, меньше платят физические лица.

Q: Брать ли деньги за новые версии со старых клиентов?

A: Это во многом зависит от принципа работы программы. В частности, если это антивирусная программа, то можно продавать не саму программу, а возможность обновления базы вирусов. Также можно требовать плату только при "большой" смене номера версии, скажем, при переходе с четвертой на пятую. За промежуточные версии денег не брать. Для небольших проектов более целесообразен такой подход, при котором пользователь платит за программу лишь один раз, все модификации и новые версии получая бесплатно.

Q: Что такое ордер (order)?

A: Ордер - это просто покупка. Когда говорят "мне пришел ордер", это означает, что регистратор прислал тебе письмо с реквизитами покупателя, который уже оплатил свою покупку.

Q: Что такое чарджбэк (charge-back)?

A: Charge-back (англ.) - возврат платежа. По сути дела это "насильственный" отъем денег у продавца. Владелец карты просто идет в свой банк и инициирует процедуру "чарджбэк". При этом с момента покупки товара может пройти весьма значительный срок (бывает даже порядка полугода), но банк, как правило, становится на сторону своего клиента и удовлетворяет его просьбу. Однако чем больше времени прошло с момента покупки, тем с меньшей охотой банк идет на эту операцию. Далее банк плательщика уведомляет банк твоего регистратора и принудительно забирает у него причитающуюся сумму, регистратор же в свою очередь списывает ее с твоего счета, да еще и штрафует тебя. Очень неприятная процедура, тем более что если у тебя будет много чарджбэков, регистратор может просто расторгнуть с тобой договор обслуживания – им лишние проблемы не нужны.

Q: Что такое рефанд (refund)?

A: Это способ полюбовного решения проблемы возврата денег покупателем. Обычно рефанд происходит так: покупатель пишет тебе, что "понимаете, программа не совсем меня удовлетворила, и я хочу вернуть деньги". Тут ты можешь сказать ему, что ты принципиально не осуществляешь рефанды (был же тральный период, человек мог спокойно во всем разобраться), но в этом случае ты рискуешь нарваться на чарджбэк. Или ты обращаешься к своему регистратору, и он спокойно возвращает

деньги покупателю, при этом никаких штрафов не предусмотрено.

Q: Что такое аффилиат (affiliate)?

A: Это посредник между тобой и покупателем. Как правило, это владелец файлового архива, который в случае если ты его запишешь в свои аффилиаты, будет как-то продвигать твою программу на своем сайте. Аффилиатам при регистрации выдаются специальные ссылки на страницу покупки твоей программы. Для покупателя вся схема прозрачна: он думает, что покупает у тебя, однако в действительности тебе достанется в лучшем случае 70% от сделки, так как 30% отойдут аффилиату.

Q: Что такое ключевые слова?

A: Ключевыми называются те слова, по которым твой сайт и, соответственно, твою программу должны находить люди, использующие поисковики. Обычно продвижение сайта начинается именно с подбора ключевых слов. В частности, для твоего сайта, продающего программу, ключевыми словами будет название программы, ее класс (например "браузер", "текстовый редактор", "почтовый клиент"), ключевые характеристики ("конвертирование графики", "распаковка архивов"). Грамотный подбор ключевых слов - залог успеха в продвижении сайта.

Q: Что такое контекстная реклама?

A: Это реклама, появляющаяся в результатах поиска поисковой машины. Контекстную рекламу целесообразно покупать в том случае, если твой продукт или сайт не попадает в результаты поиска по важным для тебя ключевым фразам. Оплата за рекламу обычно осуществляется в соответствии с фактом перехода посетителя на твой сайт, то есть за каждый клик по твоему рекламному объявлению.

Q: Что такое SEO?

A: Search Engine Optimization - оптимизация содержимого сайта таким образом, чтобы по интересующим тебя ключевым словам твой сайт был на первой странице результатов поиска. На самом деле это очень большая отрасль IT-технологий, в которых тебе предстоит разобраться. Для получения более подробной информации советую заглянуть сюда: http://forum.searchengines.ru.

Q: Что такое кастоминг и кастом-версия?

A: Кастоминг - это доработка программного продукта под требования конкретного заказчика, в результате которой получается кастом-версия. Обычно подобные версии просят аффилиаты, чтобы точно отслеживать, купил ли человек программу и воспользовался ли он их ссылкой или ссылкой автора программы.

Q: Что такое позиционирование программы и для чего оно нужно?

A: Позиционирование - это размещение программы на определенной рыночной нише. Проблема обычно заключается в том, что если программа не совсем точно соответствует нише, приходится исхитряться с ее преподнесением людям. Позиционирование очень важно в продвижении твоей программы, так как один и тот же продукт при разном качестве позиционирования может достичь совершенно разных уровней продаж, а при плохом позиционировании может вообще ничего не достичь. Поэтому, чтобы не ошибиться в самом начале своей работы, рекомендую присмотреться к конкурентам: как они позиционируют свои программы, на чем они делают акценты в рекламе продуктов, на что обращают внимание покупателей, чем мотивируют его к покупке.

Q: Что такое adwords?

A: Так называется проект Google, предназначенный для продажи контекстной рекламы для этого поисковика. Эта система особенно хороша тем, что начинает работать почти сразу после того как ты перевел им деньги и все настроил. При этом очень демократично построена система оплаты, ты можешь задать бюджет своей рекламной кампании: чтобы в день было представлено рекламы не больше, чем на сумму, установленную тобой. Хочу обратить твое внимание на то, что в этой системе строго ограничено количество слов в объявлении, в котором всего три строчки, вмещающие всего около сотни символов, значит, такое объявление должно быть очень "сильным" и мотивирующим.

Q: Что такое индекс цитирования - PageRank?

A: Индекс цитирования применяется поисковиком Яндекс и представляет собой число от нуля до практически бесконечности. Это число, которое определяет количество уникальных доменных имен (сайтов), на которых стоит ссылка на твой сайт. PageRank применяет поисковик Google. По сути дела, это тот же индекс цитирования, только его значение может быть от нуля до десяти. Увеличение индекса цитирования (и PageRank'а) приводит к тому, что по ключевым словам в результатах поиска твой сайт будет выше аналогичных страниц с меньшим индексом цитирования. Вот почему за эти показатели нужно бороться всеми законными способами.

Q: Что такое сабмит и для чего он нужен?

A: Сабмит - это размещение информации о твоей программе в файловом архиве. Это очень важный момент раскрутки, ему необходимо уделить особое внимание.

Q: Что такое дорвей?

A: Дорвей - это обычная на вид web-страница, которая под завязку напичкана ключевыми словами. Это делается специально для того, чтобы поисковик пометил ее как очень "крутую" и чтобы она всегда была в верху результатов поиска по ключевым словам, написанным на ней.
Все это звучит прекрасно, но если ты решился сделать несколько дорвеев, то бойся поисковика - может раскусить тебя и забанить. То есть поисковик может удалить дорвей его из своей базы, и, как правило, в таком случае обратной дороги не остается. Важно учесть такой простой факт: в тексте страницы количество повторений каждого ключевого слова не должно превышать 5-10% от объема всего текста, то есть если у тебя текст на 1000 слов, в них будет нормально включить лишь десяток ключевых слов.

Александр Лозовский (alexander@real.xakep.ru)

STADYJOB ДЛЯ КОДЕРА

КАК НАЙТИ РАБОТУ ПРОГРАММИСТОМ

В нашей стране, несмотря ни на что, много хороших программеров. Нередко встречаются и очень хорошие. Поэтому мне странно смотреть, как эти хорошие и очень хорошие кодеры сидят дома, учатся в подозрительных ВУЗах и подрабатывают в "Макдоналдсе", мотивируя это тем, что "найти работу сложно", "кто меня возьмет", "кому я нужен без стажа".

На самом деле любой пассивный стиль поведения - это порочная тактика :). Как говорится, жить значит бороться, поэтому да здравствует поиск работы по знаниям.

Можно действовать двумя способами. Первый - это "работа на себя", то есть шароварные программы и прочее творчество (кто сказал "спам, кардинг, порно"? Выйти из зала. Это приличная статья). Скорее всего, поначалу заработать этим не получится. Даже для опытных программеров это не более чем второй источник дохода, а то и просто хобби. Второй способ – это работа на "того дядю", поиском которой мы и будем заниматься прямо сейчас.

Искать работу на дядю можно тучей разных методов:

По знакомым. Есть знакомые программеры? Как они устроились? Куда? Может, и я смогу? Знакомый организовал фирму? Какая же фирма - и без программера? Обоснуй необходимость его наличия в штате. Поначалу программер может просто объяснять, почему принтер не печатает и что делать, если курсор хочется двигать дальше, а коврик уже закончился.

По форумам. Толкаться по кодерским форумам и постгруппам – не только трата времени, но и а) бесплатная помощь в кодинге; б) бесплатная помощь от тебя окружающим и, как следствие, их глубокий респект. А что такое респект? Респект – это когда в один прекрасный момент ты запостил на этом форуме в "Барахолке" или в другом соответствующем месте свое резюме: многие тусовщики форума не только обратят на тебя внимание, но и поспрашивают знакомых, не нужен ли кому добрый и ответственный мегапрограммер. А чтобы они догадались поспрашивать, не забудь намекнуть на это в своем посте.

По сайтам работ. Да, я имею ввиду job.ru или буржуйский www.jobpilot.com и им подобные. У меня есть достаточно знакомых, которые смогли устроиться таким образом. Не поленись сунуть туда свое однажды написанное резюме, copy&paste – не очень сложная процедура. Есть только одна проблема. Если ты пишешь о себе, то писать "Образование неоконченное высшее, МГТУ им. Баумана, C++, STL - хороший уровень, могу разобраться в чужом коде. Английский язык – в школе учил. Опыт работы – 0,000" не очень хорошо, потому что работодателю обычно нужно:

❶. Хороший программерский скилл (это у тебя есть).

❷. Умение работать в команде кодеров, а именно опыт работы. Если его не имеется, не будем акцентировать на этом внимание.

❸. Знание английского хотя бы для чтения мануалов в оригинале. Лучше в совершенстве. Это не только круче, но и может помочь устроиться на телеработу в буржуинскую контору.

❹. Умение разбирать чужой код. Обычно если ты приходишь в контору, перед тобой выкладываются начинания твоего предшественника. А теперь – пристегнись, потому что существуют конторы, до сих пор не вышедшие из XX века. Разбор, дописывание и апгрейд программ, сделанных еще при царе Горохе на давно вымершем языке программирования – дело трудное, а написать все заново обычно невозможно. Слишком много уже сделано, делать заново – угробить не один год. Бывает, что узрев такую благодать, программеры просто бегут из такой конторы на следующий же день работы :). Обычно распознать такую засаду можно по наличию в требованиях к соискателю знания этого языка. Сложнее бороться с чужими творениями, написанными на нормальном языке, но через одно место. Вернее, с этим не надо бороться, с этим надо жить.

Помимо резюме на сайте также не забудь оставить контакты, e-mail и мобильный телефон (лучше и то и то, многим звонить проще, чем писать) и указать примерный размер зарплаты, на которую ты рассчитываешь. Сколько запросить – думай сам исходя из своих знаний.

По сайтам компаний. Весьма благое это дело, поскольку обычно там публикуются вакансии вместе с требованиями и приблизительной зарплатой. Например, я, пробежавшись по сайтам своих любимых контор, обнаружил, что "Акронису" (acronis.ru) требуется довольно много программеров (в том числе и GUI-программист, возможно, и без опыта работы ;)), а по слухам – можно попробовать свои силы в лаборатории Касперского и в "Агнитуме" (которая выпускает Outpost Firewall).

Помимо резюме на сайте также не забудь оставить контакты, e-mail и мобильный телефон.

ПРЕДСТАВЛЯЕМ СЕБЯ

Вне зависимости от того, каким путем ты нашел вакансию, остается еще полпути: а) доказать работодателю, что ты – это то, что ему нужно; б) получить заветную запись в трудовой книжке (или без нее, как повезет).

Первый шаг – или звонок, или письмо в компанию. Что из предложенного выбрать, немаловажно: если ты законченный гик, при личном общении заикаешься и не можешь внятно излагать свои мысли (или забываешь-путаешься-стесняешься), лучше написать письмо. Но учти, что для оформления писем обычно есть определенные правила.

Например, многие компании требуют, чтобы письма были оформлены с указанием темы письма и чтобы данные излагались в определенной последовательности. Соблюсти эти правила важно, потому что даже странно написанная тема "хочу работать" вместо предписанного "вакансия программиста" если не отправит письмо в треш (автоматическая сортировка писем по теме: вся масса писем, отправленных на info@company.com, сортируется для отправки разным сотрудникам, а письма, не подходящие ни под какой шаблон, отправляются в РАЗНОЕ, на которое всем забить болт), то, как минимум, скажет кое-что о внимательности и дисциплинированности пишущего :).

В резюме пиши все. Нечего сказать? Только три курса МГТУ и куча написанных just for fun freeware-программ? А как насчет сертификатов? А не модерировал ли ты популярный кодерский форум? Не писал ли статьи на сайты? А может, и сам содержал популярный ресурс? Об этом тоже надо написать. Стеснительность в данном случае – зло, поэтому сконцентрируйся на описалове всего, что ты когда-либо делал. Какие технологии использовал, в чем их новизна, полезность, оригинальность. Что отличает их от собратьев по классу (маленький размер, быстродействие etc.). Не надо акцентировать внимание на том, что работу ищет студент :). Мало ли кто ее ищет. Тем более что куча ныне серьезных программеров закончили ВУЗы, которые имеют лишь отдаленное отношение к кодингу.

Чтобы лучше представить свои могучие и добытые самостоятельно знания, их можно задокументировать. Например, любители продуктов от 1С могут сдать экзамен 1С: профессионал (стоит около 20$), фанаты мегакорпорации - получить кучу статусов от Microsoft, но если честно... все это совершенно не обязательно (1С – исключение, на работу с ее продуктами несертифицированных обычно не принимают). Иногда даже в солидную контору могут взять после заполнения анкеты (правда, довольно подробной) и устного собеседования (достаточно сложного, но знания же у тебя есть? :)). Без какого-либо в/о. Вот цитата одного из программеров лаборатории Касперского: "Откуда у меня дипломы? У меня ж нету в/о... Просто на три месяца испытательный срок сделали, да и все". Не все так сложно. Кстати, чтобы не быть голословным, я взял в руки свой старенький телефон и накрутил диск в следующие конторы: в российское представительство Symantec (оказалось, программеры ему не нужны, и мне предложили приехать в штаб-квартиру (California), где они нужны. Это далековато), знакомому начальнику из фирмы "Мактор". Написал письмецо в "Агнитум" (не ответили). Результаты моего общения – на врезке. Удачи.

■ Станислав Нижниченко, заведующий отделом системного администрирования и сопровождения баз данных ООО "МАКТОР" (на Скорой помощи работает, кстати – прим. Dr.)

XS: Представь, что ты простой программер. Нужно найти работу. Как будешь искать? С чего начнешь? Что предпочтешь?
SN: Начну с того, что куплю газету с объявлениями о найме на работу и буду спрашивать у друзей, нет ли где-нибудь вакансий. Если работа не находится сразу, то можно устроится на что-нибудь околопрограммерское (эникей) или в компьютерный салон (опыт + знание железа).

XS: Ты нашел вакансию на сайте фирмы. Как составишь письмо? Как составишь резюме, если там нет готовой формы?
SN: У меня :) есть резюме, составленное на русском и английском языках, но если бы его не было, то я купил бы книжку по составлению резюме! Благо их на рынке много. Это я говорю с учетом того, что не имею денег на разного рода интернеты и прочие блага, доступные за ЛИШНИЕ деньги.

XS: Ты нашел что-то и хочешь устроиться на работу к такому злому начальнику, как ты сам :). Как будешь вести себя? Как докажешь, что ты - именно тот человек, который нужен? Будешь ли демонстрировать знания (если будешь, то как?) или будешь ждать, пока спросят?
SN: Что-то доказывать стоит после того как тобой заинтересовались - это самое главное. Дальше можно помочь ему решить текущие проблемы :), показав таким образом на практике свои знания в данной области и т.д. и т.п. Далее просто разговор, поиск общих интересов (специфика нашей страны).

XS: Какие качества в программисте ценишь больше всего? Каким должен быть идеальный программер с точки зрения начальника? :)
SN: Нестандартное мышление, способность поправить меня, если я ставлю задачу, то есть ухватить суть проблемы и начать решать ее. А если еще и предложит несколько способов решения и позволит мне выбрать, то это просто супер! Ну и плюс опять же специфика страны: чтобы человек был хороший и соблюдал интересы мои и фирмы.

XS: Допустим, к тебе пришел студент-технарь лет 20-25. Знания соответствуют твоим требованиям. Возьмешь?
SN: По поводу студентов есть некоторая неоднозначность, как то: допустим, есть некая срочная работа и я поручаю ему выполнить эту работу, а тут у него семестр заканчивается и все отсюда вытекающее. Правда, если человек на последних курсах, то таких неприятных ситуаций не возникает.

XS: Как относишься к сертифицированным программистам? Какой бумагой тебя можно поразить? :)
SN: Если человек собирается заниматься программизмом, то наличие сертификатов по языку (C\C++ или по еще какому-нибудь) только повысит мой интерес к этому кандидату. Но это должен быть не просто сертификат, а сертификат, подкрепленный знаниями и программами. Наверное, так.

СПЕЦ АНОНС

Читай в следующем номере Спеца

ЦИФРОВОЕ ВИДЕО

- Теория цифрового видео
- Форматы и методы компрессии
- Профессиональнык видеокарты
- Устройство видеокамеры, обзор и выбор
- Плагины для обработки видео
- Видеохудожники и видеоарт
- iVideo - монтаж видел на Mac
- Домашняя видеостудия от A до Я
- Оцифровка видео
- Твой первый фильм в Adobe Premiere
- Практика видеомонтажа
- Спецэффекты
- Адекватное аудио
- Видео во флеше
- Как снимать - практика

Весь софт на CD!

А также:

- СРАВНЕНИЕ КОДЕКОВ, ТЕСТИРОВАНИЕ ПЛАТ ВИДЕОЗАХВАТА И ЕЩЕ ЦЕЛОЕ МОРЕ ПОЛЕЗНОЙ ИНФОРМАЦИИ!

СКОРО В СПЕЦЕ:

● ВЗЛОМ И ЗАЩИТА ПРОГРАММ
Методы взлома программ. Дизассемблирование, отладка, dumping. Реализация и снятие защиты. Шифрование и сжатие, упаковка. Восстановление таблицы импорта. Защита. Вирусные технологии для защиты от cracking'а. Низкоуровневая и аппаратная защита

● МОБИЛЬНЫЕ УСТРОЙТВА И ИХ БЕЗОПАСНОСТЬ
Взлом с помощью мобильных устройств. Bluejacking, bluesnarfing и взлом Wi-Fi-сетей. Сниферов Wi-Fi\Bluetooth. Все о wardriving. Мобильные вирусы и трояны. Security-софт под мобильные платформы. Фрикинг, безопасность в телекоммуникациях. Спам.

● ИНТЕРНЕТ-ДЕНЬГИ
Обменники валюты, казино и другие web-сервисы, связанные с интернет-валютой. Различные системы: WebMoney, e-gold, GoldMoney, PayPal g.p. Заработок\процессинг: что и как реализовать. Как сделать свою пирамиду\банк, как кидают в e-бизнесе.

● СКРЫТАЯ УГРОЗА
Большой брат следит за тобой! Шпонские хитрости. СОРМ и ее аналаги в других странах, обход этих систем. Жучки и другое шпионское оборудование: как обнаружить, как собрать. Компьютерный шпионаж. Тайна PGP. Секреты реализация слежки, противостояние ей

● КОМПЬЮТЕРЫ БУДУЩЕГО
Каким будет компьютер через 30-50 лет. Технологии: квантовые, нейрокомпьютеры, языки программирование, криптография и хакеры будущего. Наноэлектроника, биотехнологии: современные достижения, что есть уже сейчас. Все о компьютерных имплантатах в человеческое тело. Интеграция человека и компьютера.

MAXIMUM ACTION! MAXIMUM BIKE!

УЖЕ В ПРОДАЖЕ

ЛУЧШИЙ ЖУРНАЛ О МАУНТИН БАЙКИНГЕ

MOUNTAIN BIKE ACTION

(game)land

d()c (doc@nnm.ru)

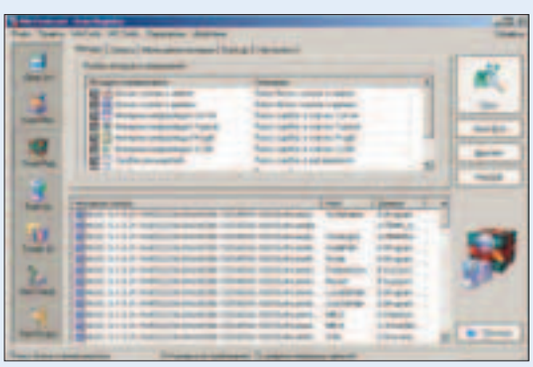
СОФТ ОТ NONAME

NERO INFOTOOL 3.0

Свежий релиз. Несмотря на небольшой размер, программа показывает весьма подробную информацию о CD/DVD-приводах: максимальную скорость чтения и записи данных, размер буфера, поддерживаемые возможности, информацию о вставленном диске, прочую информацию о железе.
Автоматически находит и показывает софт для работы с CD. Программка бесплатна и пригодится всем, кто хочет знать больше!

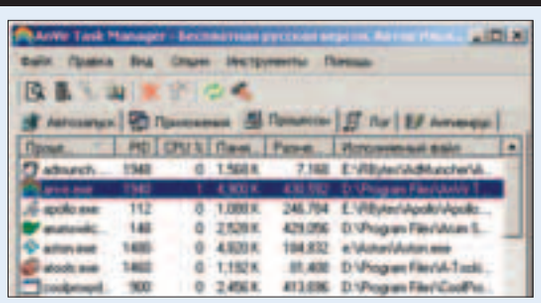
WINTOOLS.NET CLASSIC 5.3

Набор инструментов, предназначенный для поддержания операционной системы в нормальном состоянии на протяжении всего эксплуатационного периода и для ускорения ее работы.
В комплект поставки входят инструменты для корректного удаления программ, для слежения за изменениями, происходящими в процессе инсталляции; очистки жесткого диска от временных файлов; реестра Windows от устаревших записей; а также менеджер автозагрузки и небольшой твикер, позволяющий управлять некоторыми скрытыми настройками и возможностями операционной системы, влияющими на безопасность, производительность и удобство использования системы.



ANVIR TASK MANAGER 3.7

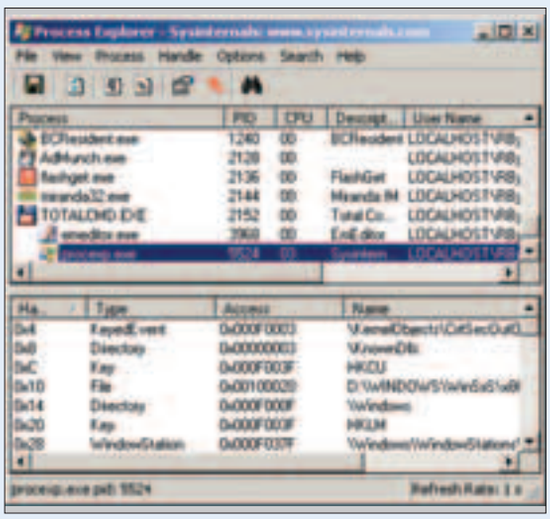
...А если ты не разработчик, то тебе подойдет этот инструмент. Он позволяет все то же, что умеет стандартный (изменять приоритеты, завершать и т.д.), плюс быстро просматривать свойства запущенного файла, его физическое расположение (запросто можно определять, куда скопировал себя вирус/троян), добавлять к заблокированным, показывать загрузку процессора (а в подсказке еще и "тяжелые" процессы) и многое другое.
Несмотря на то, что удобный Task Manager - основная функциональная часть программы, в ней присутствует и удобный менеджер автозапускаемых при старте Windows приложений и лог, в котором показывается, кто какое приложение запустил. У программы есть и еще одна функция - слежение за автозапускаемыми файлами: если какая-то



программа пропишет себя в реестр для запуска при старте Windows, AnVir Task Manager всегда предупредит тебя об этом!

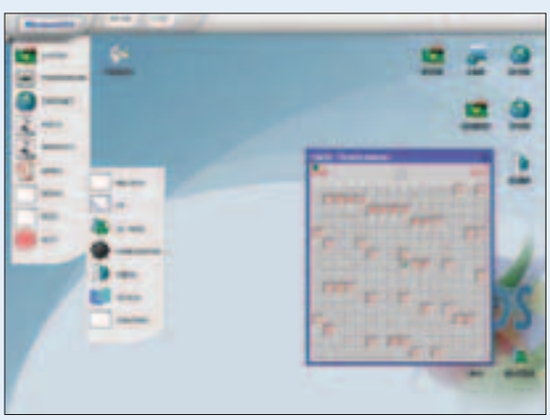
PROCESS EXPLORER 9.01

Небольшая бесплатная программка менеджер процессов. Показывает детальную информацию о каждом процессе, в том числе статистику памяти, атрибуты безопасности, используемые DLL и многое другое. Естественно, каждый процесс можно прибить, изменить приоритет и просмотреть другую информацию. Есть возможность поиска. В целом Process Explorer - штука мощная, но для повседневного использования подходит слабо, впрочем, для разработчиков ПО это идеальный инструмент.



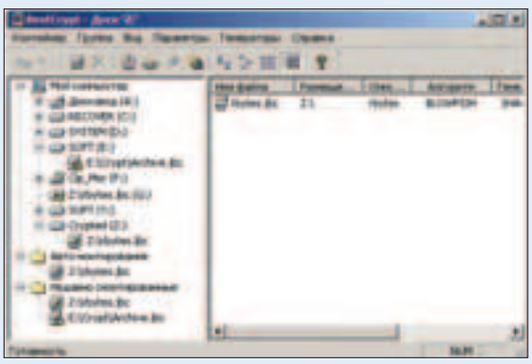
MENUETOS 0.78 PRE6

Полноценная операционная система, написанная на ассемблере, с набором основных драйверов и встроенным в ядро GUI. Для работы может использовать существующий раздел FAT32 (а может и вообще не использовать HDD, см. ниже). Имеет встроенные ftp/http/mp3/smtp-серверы; irc, http, nntp и tftp-клиенты, средства разработки.
GUI держит разрешения до 1280x1024. Одно из достоинств в том, что все помещается на одной дискете. Конечно, для нормальной работы нужно подыскать что-то другое, но просто посмотреть стоит :). Достаточно загрузиться с образа дискеты в эмуляторе VMWare или Virtual PC. На официальном сайте (www.menuetos.org) доступны также исходники. Более подробную информацию на русском (включая русификатор) можно получить на сайте http://menuet.narod.ru.



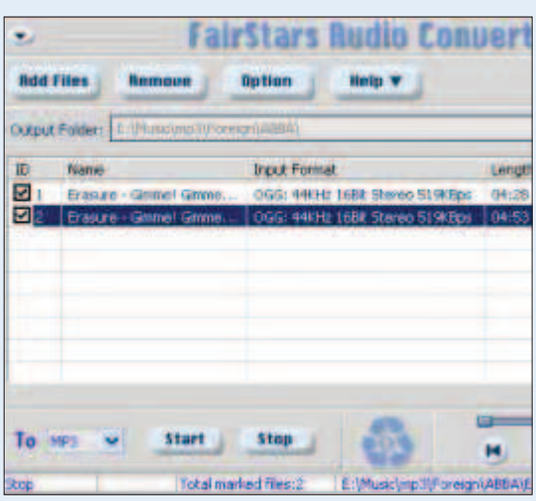
BESTCRYPT 7.12.01

Программа, создающая шифрованные диски. Диски ничем не будут отличаться от реальных: каждому можно назначить свою букву (из свободных), только данные будут храниться в зашифрованных файлах-контейнерах.
Поддерживаются алгоритмы Blowfish, Twofish, Rijndael, GOST 28147-89, количество файлов-контейнеров не ограничено. Доступ к шифрованному диску можно будет получить только введя пароль, то есть если у тебя вдруг появится что скрывать, то BestCrypt придет к тебе на помощь :).



FAIRSTARS AUDIO CONVERTER 1.50

Отличнаая программа, которая сочтет за счастье сконвертить для тебя какой-нибудь звуковой файл из одного формата в другой. Программа прошла отличную профподготовку, поддерживает более двух десятков форматов, причем не просто тупо поддерживает их, но и понимает кое-какие опции кодирования. Работает шустро, при необходимости пакетно, временных файлов не делает, а шпарит напрямую. По твоему велению запросто приведет громкость файлов к общему знаменателю (то есть нормализует). Также можно попутно ID3-теги поредактировать. По окончании работы вполне в состоянии воспроизвести плоды собственного труда. Единственный минус, который я пока обнаружил в процессе эксплуатации этой софтинки, - во время конвертирования файлов она загружает CPU на 100%, что отнюдь не способствует выполнению параллельно каких-либо других дел. Но на фоне общего впечатления о программе это уже мелочи :).



Content:

Сергей Никитин, Дмитрий Шамаев, test_lab (test_lab@gameland.ru)

НОВЫЙ СРЕДНИЙ КЛАСС

ИЩЕМ ЗОЛОТУЮ СЕРЕДИНУ СРЕДИ PCI-E ВИДЕОАКСЛЕРАТОРОВ

Новая графическая шина PCI-Express x16 и устройства, рассчитанные для работы с ней, стали доступны. Их много, на любой вкус и кошелек. Как всегда, фантастически мощные платы-монстры, взращенные на самых передовых технологиях и имеющие гигантские коробки с богатейшим комплектом поставки, набирают тысячи баллов в тестах и сотни fps в играх, но совсем не радуют нереально высокими ценами.

Есть и другая крайность – дешевые, самые дешевые платы под новую шину. Слабая производительность уровня устройств предыдущего поколения – их основной недостаток. Поэтому нам нужна золотая середина – не самые дорогие платы, обеспечивающие хорошую скорость в играх. Таких сейчас много, и мы о них расскажем.

test_lab выражает благодарность за предоставленное оборудование компании ISM Computers (тел. (095) 956-9377, www.ism.ru), а также российским представительствам компаний Asus, ATi, Gigabyte, NVIDIA, Sapphire.

СПИСОК УСТРОЙСТВ

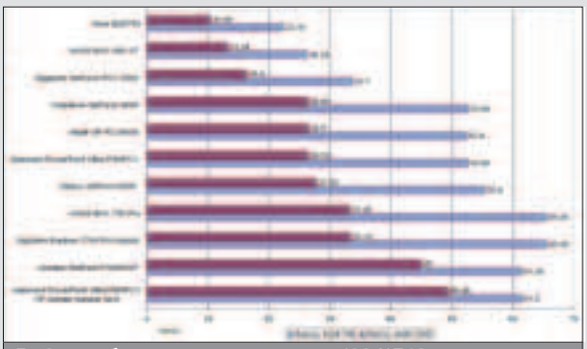
	ASUS EAX 700PRO
	Gainward 6600GT
	Gigabyte GeForce PCX 5900
	Galaxy GeForce 6600
	Leadtek WinFast PX 6600GTD-Link DSL-300T
	ASUS EAX 600XT
	Manli GF PCX 6600
	ASUS EN5750
	Sapphire Radeon X700Pro Hybrid
	Chaintech GeForce 6600
	Gainward Ultra\1740PCX (GeForce 6600)

МЕТОДИКА ТЕСТИРОВАНИЯ

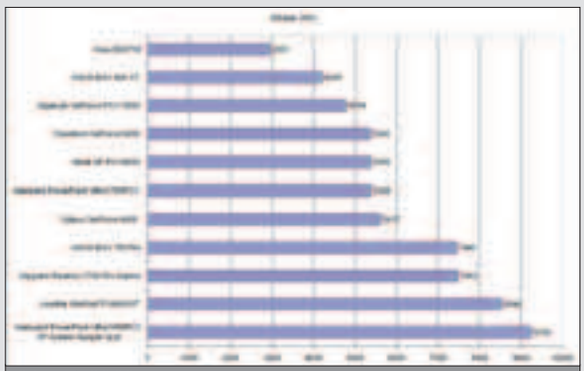
Перед началом тестирования был отформатирован жесткий диск и установлена операционная система Windows XP SP2. Для видеоплат на чипах ATI использовалась последняя версия драйверов ATI Catalyst 5.2. Для плат, в основе которых графические процессоры производства NVIDIA, была использована последняя официальная бета-версия драйверов Detonator 71.84. По наличию игр и другого ПО оценивался комплект поставки видеокарты. Уровень шума определяли по своим субъективным слуховым ощущениям во время проведения тестов при принудительной остановке процессорного кулера. Производительность измерялась как в синтетических тестах (3DMark 2001SE, 3DMark 2003 патч 360, 3DMark патч 120 с установками по умолчанию), так и в реальных игровых – FarCry 1.3, Doom 3, Half-Life 2. Во всех игровых приложениях тесты проводились в двух разрешениях 1024x768 и 1600x1200 при максимальной детализации. Для уменьшения погрешности измерений игры прогнали по два раза во всех режимах, а в окончательный результат вошло их среднеарифметическое значение. В Doom 3 и Half-Life 2 использовались специально записанные нами демо, причем в Half-Life 2 мы использовали две демо с разных уровней.

Тестовый стенд
Материнская плата: Asus P5GD1
Процессор: Intel Pentium 4 550 (3,4 ГГц, Prescott)
Память: 2x512 Мб DDR400 Hynix Original
Кулер: Zalman CNPS 7700 Cu
HDD: Seagate ST320822AS
БП: 420 Вт PowerMan Pro

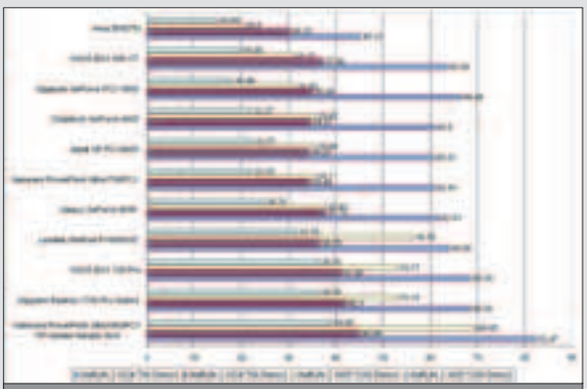
3DMark 2001SE: один из немногих тестов, где выигрывают модели на ядре ATI Radeon X700 Pro



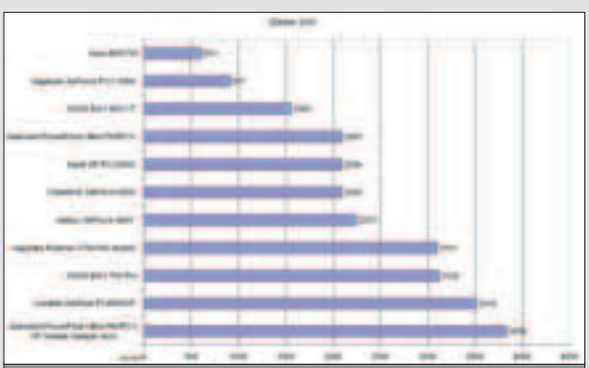
*FarCry: на более низком разрешении 1024*768 лидирует группа плат на чипах ATI, но если увеличить разрешение до 1600*1200, то в лидеры попадут те, кто сделал ставку на NVIDIA*



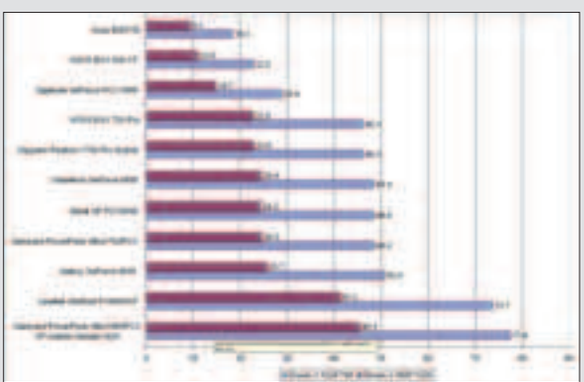
3DMark 2003: немного увеличенные частоты ядра и увеличенный объем памяти не дали ощутимого прироста для Galaxy GeForce 6600



*HalfLife 2: неожиданно высокий результат в разрешении 1024*768 показала Gigabyte GeForce PCX 5900 128 Мб*

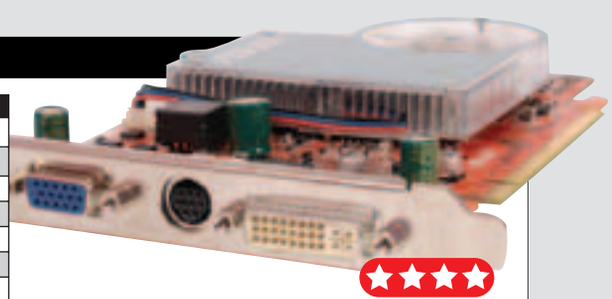


3DMark 2005: цена аутсайдера теста ниже цены лидера менее чем в два раза, а разница в производительности - более чем в шесть раз (!)



Doom 3: NVIDIA в этом тесте является безусловным лидером

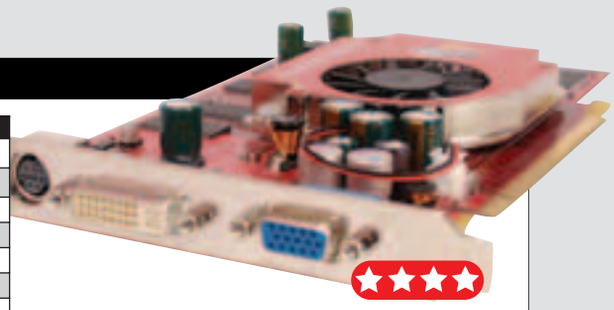
ASUS EAX 700 PRO



Эта плата построена на одном из самых мощных чипсетов в нашем обзоре - ATI Radeon X700 PRO, к тому же она имеет 256 Мб быстрой памяти GDDR 3. Такой тандем дал очень предсказуемый, но несмотря на это приятный результат - высокую производительность. Тот, кому покажется мало, может при наличии определенных навыков и умений увеличить производительность путем разгона, что вполне реально сделать до уровня скорости XT-чипа. Внешний вид кулера отличается от наружности плат на этом чипе, дизайн которых референсный. Кулер (как и стандартный) шумит, но все-таки меньше. Любителям видеотворчества придется по вкусу то, что порт VIVO живет на плате вместе со всеми необходимыми проводами в комплекте поставки. Поддержка шины PCI-E обеспечивается напрямую, а не через мост. Комплект поставки очень беден: кроме проводов для VIVO в нем, по сути, ничего и нет. Игр нет вообще, и это печально. А цена платы довольно высока. Вообще, судя по нашему тестированию, ASUS в чем-то меняет свою политику по отношению к комплектации плат. Дополнительное питание плате не требуется: все, что нужно, поставляется через шину PCI-Express.

Технические характеристики:
Ядро: ATI RV410
Количество пиксельных конвейеров, шт: 8
Шина памяти, бит: 128
Объем памяти, Мб: 256
Частота ядра, МГц: 425
Частота памяти, МГц: 432 (865)
Тип памяти: GDDR-3
Латентность памяти, нс: 2,0
Техпроцесс ядра, мкм: 0,11
VIVO: есть
ПО в комплекте: отсутствует
Цена: $270

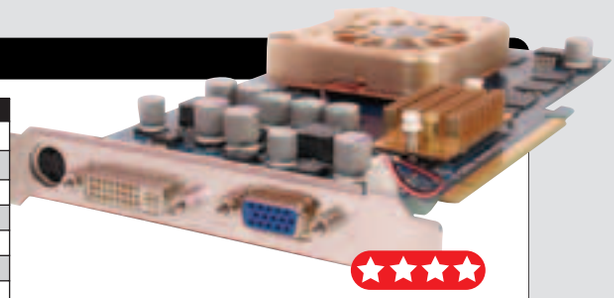
GAINWARD POWERPACK ULTRA1760PCX 128МБ



Технические характеристики:
Ядро: NVIDIA NV43
Количество пиксельных конвейеров, шт: 8
Шина памяти, бит: 128
Объем памяти, Мб: 128
Частота ядра, МГц: 300
Частота памяти, МГц: 275 (550)
Тип памяти: DDR1, TSOP
Латентность памяти, нс: 3,6
Техпроцесс ядра, мкм: 0,11
VIVO: нет
ПО в комплекте: отсутствует
Цена: $129

Всем известно, что красный - это цвет страсти, скорости и агрессии. Автомобили Ferrari, славящиеся своими динамическими характеристиками, выкрашивают в красный цвет, быки бросаются на красные тряпки. Видимо, чтобы навеять потребителям такие ассоциации, компания Gainward выбрала этот цвет текстолита для своей платы, которая оснащена мощным графчипом, работающим на повышенных частотах, и памятью GDDR3 с латентностью 1,6 нс (жаль только, что 128-ю, а не 256-ю Мб).Высочайшая производительность - главное достижение разработчиков этой платы. Кулер на плате имеет дизайн, измененный по сравнению с референсным, и светится опять же красным цветом, но сильно шумит. Радиаторы наклеены не только на процессор, но и на память. Тем, кого не устроит базовая скорость, поможет фирменная разгонная утилита, а маньяки с толстыми кошельками смогут установить ее в режим SLI. Для тех, кого интересует не только скорость, на плате есть порт VIVO, а в комплекте поставки находится соответствующее ПО. Так что творческим натурам будет где развернуться. Плата не имеет устаревших аналоговых портов D-SUB, только два цифровых DVI (правда, с переходниками), и это лишний раз подтверждает ее полный технический и технологический авангардизм.

GIGABYTE GEFORCE PCX 5900 128 МБ



Технические характеристики:
Ядро: NVIDIA NV35
Количество пиксельных конвейеров, шт: 8
Шина памяти, бит: 256
Объем памяти, Мб: 128
Частота ядра, МГц: 350
Частота памяти, МГц: 275 (550)
Тип памяти: DDR-1, BGA
Латентность памяти, нс: 2,8
Техпроцесс ядра, мкм: 0,13
VIVO: нет
ПО в комплекте: SpellForce The Order of Dawn 2CD, Rainbow Six 3 Raven Shield 2CD, CyberLink PowerDVD 5
Цена: $220

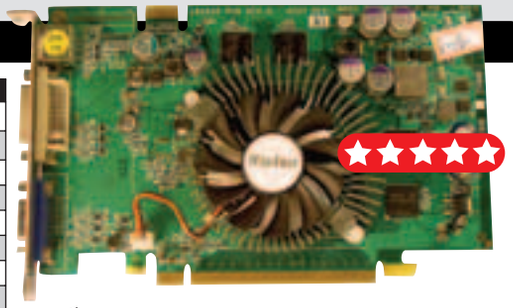
Знаешь поговорки "Кто старое помянет - тому глаз вон" и "Все новое - это хорошо забытое старое"? К этой плате относится только вторая поговорка, потому что плата собрана на чипе GeForce FX 5900. Чтобы не отставать от ATI, которая очень быстро выпустила PCI-Express платы всех ценовых уровней, и не отдать ей рынок middle-end видеокарт, NVIDIA сделала ход конем: переделала старый чипсет под новую шину (новых HMC среднего уровня для PCI-E у нее до недавнего времени в ассортименте не было, 6200 появился позже). Эта плата оснащена 256-битной памятью в корпусе BGA, что означает повышенную частоту ее работы, уменьшение нагрева и электропотребления. Плата гонится очень неплохо и имеет хороший комплект поставки. Но, несмотря на рульную память, старость чипсета дает о себе знать - все-таки предыдущее поколение. Производительность низкая, память DDR1 тоже не добавляет скорости. Это самая длинная плата в обзоре, ее кулер сильно шумит, а поддержка PCI-E организована с помощью моста, что не позволит воспользоваться теми преимуществами, которые эта шина даст в будущем. Плата имеет всего четыре пиксельных конвейера, но на закуску в комплект поставки поданы две игры: Spell Force: The Order of Dawn, Rainbow Six 3: Raven Shield.

GALAXY GEFORCE 6600 256 МБ

Технические характеристики:
Ядро: NVIDIA NV43
Количество пиксельных конвейеров, шт: 8
Шина памяти, бит: 128
Объем памяти, Мб: 256
Частота ядра, МГц: 324
Частота памяти, МГц: 276 (552)
Тип памяти: DDR1, TSOP
Латентность памяти, нс: 3,3
Техпроцесс ядра, мкм: 0,11
VIVO: нет
ПО в комплекте: CyberLink PowerDVD 5, Moto GP Ultimate Racing Technology 2
Цена: $155

В тихом омуте черти водятся. В этом устройстве, наверное, тоже. Эта плата от малоизвестного в России производителя поистине восхищает не только своей работой, но и дешевизной, так как за бренд тут много не требуют. Плата имеет на борту 256 Мб старой памяти DDR1, но другую на GeForce 6600 не ставят, зато латентность у нее самая низкая - 3,3 нс (среди плат на GeForce 6600). Вообще, по памяти и ядру этот графический акселератор гонится хорошо, но и его базовая производительность, самая высокая в нашем тесте среди решений на аналогичном чипсете, впечатляет. Единственное, что не смогло не огорчить нас - массивность и шумливость кулера. Этот недостаток частично компенсирует наличие игры Moto GP в комплекте поставки. Так что на одной чаше весов производительность, невысокая цена и разгон, а на другой - шумный кулер большого размера.

LEADTEK WINFAST PX6600GT 128 МБ



Еще одно устройство на самом оптимальном чипсете из линейки 6600. Если к этому мощному чипсету добавить память GDDR 3, то можно получить отличную производительность, что подтверждает рейтинг этого акселератора в наших тестах (второе место). Конечно, жаль, что памяти установлено всего 128 Мб и что на нее не наклеены радиаторы, но кулер очень тихий и производительный. Для удобного и безопасного разгона в комплект поставки входит специальная фирменная утилита.

Технические характеристики:
Ядро: NVIDIA NV43
Количество пикселных конвейеров, шт: 8
Шина памяти, бит: 128
Объем памяти, Мб: 128
Частота ядра, МГц: 500
Частота памяти, МГц: 500 (1000)
Тип памяти: GDDR-3
Патентность памяти, нс: 2,0
Техпроцесс ядра, мкм: 0,11
VIVO: есть
ПО в комплекте: Prince of Persia The Sands of Time 2CD, Splinter Cell Pandora Tomorrow DVD
Цена: $241

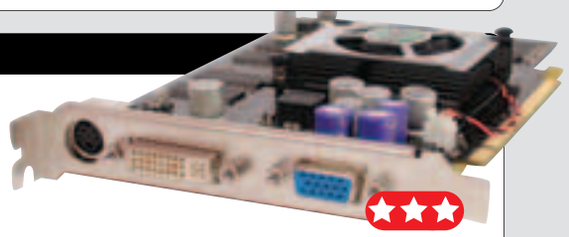
Производительность можно увеличить не только разгоном, но и более кардинальным образом - установить плату в режим SLI (эта возможность поддерживается). Так что выбор за тобой. Наигравшись на максимальных скоростях в игры, ты сможешь снять фильм о себе любимом, а дальше монтировать его на компьютере, в чем поможет порт VIVO, установленный на плате. Комплект поставки довольно богатый и помимо всего прочего содержит такие игры, как Prince of Persia: Sands of Time и Splinter Cell: Pandora Tomorrow.

ASUS EAX 600 XT 128 МБ

Плата определенно создана для тех, кто не чужд эстетике. Купер светится нежно-голубым светом, а диски из комплекта поставки помещены не в вульгарные пакетики или коробочки, а в специальный CD-кейс оранжевого цвета. Из игр обнаружены Deus Ex: Invisible War. Так как девайс оснащен портом Video-In/Video-Out, к нему прилагается и набор софта для работы с видео. Все необходимые соединительные провода также есть в комплекте поставки. Купер, хоть и прикрывает собой память, тем самым охлаждая ее, очень сильно шумит. Памяти не очень много - 128 Мб типа DDR1, что вкупе со всего четырьмя пиксельными конвейерами в устаревшем чипсете дает очень невысокую производительность.

Технические характеристики:
Ядро: ATI RV380
Количество пикселных конвейеров, шт: 4
Шина памяти, бит: 128
Объем памяти, Мб: 128
Частота ядра, МГц: 500
Частота памяти, МГц: 371 (742)
Тип памяти: DDR-1, BGA
Патентность памяти, нс: 2,0
Техпроцесс ядра, мкм: 0,13 low-k
VIVO: есть
ПО в комплекте: Deus EX Invisible War 2CD, Asus DVD XP, Asus Medi@Show SE 2.0, Asus PowerDirector 3DE, Ulead Cool 3D SE 3.0, Ulead Photo Express SE 4.0
Цена: $190

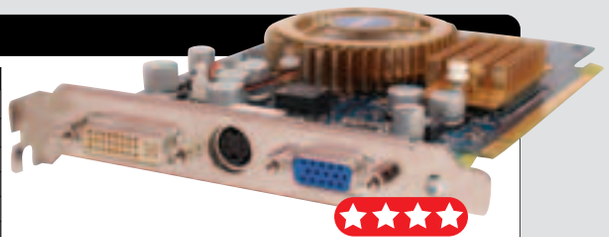
MANLI GF PCX6600 128 МБ



Это изделие имеет приемлемую цену, чему есть несколько взаимосвязанных объяснений. Во-первых, это OEM-поставка, так что не рассчитывай на то, что кроме платы обогатишься на игры, софт, провода, наклейки, переходники и т.д. Во-вторых, эта фирма у нас известна не очень широко, поэтому доля "имени" в цене не очень существенная. Плата стильно-черная, выглядит хорошо. На нее установили тихий купер с большим радиатором, который, впрочем, не прикрывает собой 128 Мб памяти DDR1. Базовая скорость средняя, но благодаря хорошему чипсету и неплохой памяти разгонный потенциал довольно высокий. Но делать его придется вручную, так как никакой утилиты для этого в комплекте поставки ты не найдешь. Повышай свои технические навыки!

Технические характеристики:
Ядро: NVIDIA NV43
Количество пикселных конвейеров, шт: 8
Шина памяти, бит: 128
Объем памяти, Мб: 128
Частота ядра, МГц: 300
Частота памяти, МГц: 275 (550)
Тип памяти: DDR1, TSOP
Патентность памяти, нс: 3,6
Техпроцесс ядра, мкм: 0,11
VIVO: нет
ПО в комплекте: отсутствует
Цена: $150

ASUS EN5750 256 МБ



Еще один представитель "рестайлинговых" (или реинженирингових?) изделий. Устройство на чипсете (NVIDIA GeForce 5700), который изначально создавался для работы с шиной AGP. Позже к нему добавили специальный переходной мост, и он стал работать на новой платформе. Что из этого получилось? Когда-то GeForce 5700 считался хорошим чипсетом среднего уровня, на этой плате установлено 256 Мб памяти DDR1 - большое количество. Однако имеется всего четыре пикселных конвейера, да и мост не позволяет в полной мере воспользоваться всеми возможностями PCI-Express, так что производительность невысока. Купер тоже не порадовал - слабенький и громкий, память не закрывает, а радиаторы на нее не установлены. Зато в комплект поставки входит игра Deus Ex: Invisible War.

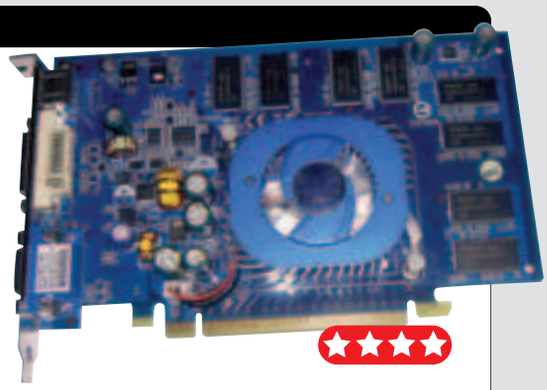
Технические характеристики:
Ядро: NVIDIA NV36
Количество пикселных конвейеров, шт: 4
Шина памяти, бит: 128
Объем памяти, Мб: 256
Частота ядра, МГц: 425
Частота памяти, МГц: 250 (500)
Тип памяти: DDR1, BGA
Патентность памяти, нс: 4,0
Техпроцесс ядра, мкм: 0,13
VIVO: нет
ПО в комплекте: AsusDVD XP, Deus EX Invisible War 2CD
Цена: $155

SAPPHIRE RADEON X700PRO HYBRID

Изделие выполнено в фирменном "синем" стиле компании Sapphire, так что фанаты будут рады, а остальные пользователи не смогут перепутать ни с чем другим. На плате установлена быстрая память GDDR3 объемом 256 Мб, что наверняка понравится играм с большими текстурами. Память не перегреется: на нее наклеены радиаторы. На графическом процессоре установлен почти бесшумный купер среднего размера. Чипсет довольно мощный, память быстрая и современная, так что производительность у этой платы очень неплохая. Можно попробовать увеличить скорость путем разгона, но перспективы при этом не самые лучшие: до тебя с платой уже поработали инженеры ATI. Комплект поставки солидный, и помимо всего прочего в него входят две хороших игры: Prince of Persia: Sands of Time и Splinter Cell: Pandora Tomorrow.

Технические характеристики:
Ядро: ATI RV410
Количество пикселных конвейеров, шт: 8
Шина памяти, бит: 128
Объем памяти, Мб: 256
Частота ядра, МГц: 425
Частота памяти, МГц: 432 (865)
Тип памяти: GDDR-3
Латентность памяти, нс: n/a
Техпроцесс ядра, мкм: 0,11
VIVO: нет
ПО в комплекте: Prince of Persia The Sands of Time 2CD, Splinter Cell Pandora Tomorrow DVD, CyberLink PowerDVD 5
Цена: $200

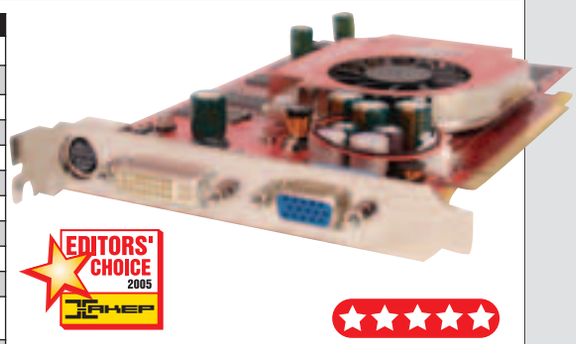
CHAINTECH GEFORCE 6600 128 МБ



Эта плата имеет странный комплект поставки - диск с демо-версиями игр и программы для работы с видео, в то время как порта VIVO на ней нет. Но это не главное. Производительность у этого изделия средняя, но хороший чипсет и 128 Мб памяти DDR1 дают неплохие шансы на разгон. Купер установлен очень достойный: его почти не слышно во время работы, но он не прикрывает память, на которую не наклеены радиаторы. Память, правда, немного подкачала в смысле латентности, которая составляет целых 4 нс (рекорд в нашем тесте). Памяти не так уж и много, всего 128 Мб. Получается, что за среднюю цену мы получаем среднюю производительность, но имеем хорошие шансы на разгон. Мы думаем, что это выгодно.

Технические характеристики:
Ядро: NVIDIA NV43
Количество пикселных конвейеров, шт: 8
Шина памяти, бит: 128
Объем памяти, Мб: 128
Частота ядра, МГц: 300
Частота памяти, МГц: 275 (550)
Тип памяти: DDR1, TSOP
Латентность памяти, нс: 4,0
Техпроцесс ядра, мкм: 0,11
VIVO: нет
ПО в комплекте: WinDVD 5, WinDVD Creator 2, WinRip 2.1, DVD Copy 2 Lite, Home Theater 2.1 Lite, Game Pack (демо-версии)
Цена: $150

GAINWARD POWERPACK ULTRA1960PCX XP GOLDEN SAMPLE GLH 128 МБ



Название платы - это не наша шутка. Сложно сказать, что подвигло Gainward дать изделию такое имя, но, видимо, причины были серьезными. Помимо зубодробительного названия плата обладает еще многими интересными особенностями. Это, например, 128 Мб памяти типа DDR1 и высокий разгонный потенциал. Правда, память не имеет радиаторов и не прикрывается кулером, который, хоть и шумный, но очень качественно и плотно прикреплен к графическому ядру, так что охлаждает его как надо. Качество исполнения на высоте. Разгон можно производить как вручную, так и с помощью фирменной утилиты из комплекта поставки, в котором, по большому счету, больше ничего и нет, бедноват он. Зато имеется порт VIVO, так что на этой плате можно не только играть, но и творить. Цена платы самая низкая в тесте.

Технические характеристики:
Ядро: NVIDIA NV43
Количество пикселных конвейеров, шт: 8
Шина памяти, бит: 128
Объем памяти, Мб: 128
Частота ядра, МГц: 540
Частота памяти, МГц: 575 (1150)
Тип памяти: GDDR-3
Латентность памяти, нс: 1,6
Техпроцесс ядра, мкм: 0,11
VIVO: есть
ПО в комплекте: muvee autoProducer 3, InterVideo WinDVD 5
Цена: $240

ВЫВОДЫ

«Выбор редакции» получает модель Gainward PowerPack Ultra/1960PCX XP Golden Sample GLH 128 Мб, победившая с приличным отрывом в 4,5 из 6 тестов за счет более высоких штатных частот относительно других видеокарт на чипе NVIDIA GeForce 6600GT.

«Лучшая Покупка» присуждается ASUS EAX 600 XT. Это проверенное временем решение, цена которого стала приемлемой. Однако, выбрав правильные настройки, ее вполне можно использовать в большинстве современных игр

Дмитрий Шамаев, test_lab (test_lab@gameland.ru)

LGA 775 С ТУРБИНОЙ

IGLOO GLACIALTECH TURBINE 4500 PRO

Если ты думаешь, что платформа Socket 478 совсем изжила себя, ты категорически неправ, потому что переход процессоров для этого разъема на новый степпинг E произошел уже после того, как они появились для LGA775. Чем может быть полезен этот степпинг? Значительно возросшим разгонным потенциалом, а значит, возможностью разгона до большей частоты, следовательно, и производительностью, которая, как ни крути, напрямую зависит от количества гига- и мегагерц. А что особенно важно при разгоне? Материнская плата?! Безусловно, она играет значительную и не менее важную роль, чем охлаждение, но именно охлаждение так важно при увеличении рабочей частоты процессора: с плохим охлаждением и даже с самой супер-пупер-материнской платой за пару сотен навечно позеленевших президентов тебе все равно не удастся сделать разгон успешным. Поэтому даже сейчас в каталогах различных компаний-производителей можно встретить новые решения для старой платформы, для той самой, на которой процессоры еще имели ноги и умели неплохо бегать. Например, не так давно появилась новая версия кулера Igloo GlacialTech Turbine 4500, которая отличается от старой всего лишь дополнительным сочетанием из трех букв в самом конце названия - "Pro". По традиции в кулерах GlacialTech линейка Pro отличается от nePro более высокими скоростными характеристиками вентиляторов при неизменном дизайне основной конструкции. Сама конструкция этой Турбины очень интересна. Радиатор состоит из четырех частей: подошвы, двух половинок и тепловых трубок. Все вместе это напоминает трубу, лежащую на подставке в горизонтальной плоскости. Две половинки этой трубы скрепляют тепловые трубки: две с одной стороны и с другой - еще одна, которая обходит по периметру весь радиатор, тем самым распределяя тепло равномерно по всей площади радиатора от самого основания до вершины. Если у конкурентов в похожих конструкциях обычно используются вертикальные или горизонтальные ребра одинаковой толщины, то в данном случае Igloo снова использует ребра игловидной формы. В этой модели это выражено особенно ярко, в основании ребра имеют толщину около 1,5-2 мм, а в вершине - около 0,5 мм, причем в самом центре никаких ребер нет, вместо них пустота, и воздух там может проходить свободно, но не идет: на этой же линии находятся моторчики двух (!!) вентиляторов, вращающихся со скоростью 3200 об/мин. У этих двух красавцев перламутрового цвета есть достоинство, которое должно особенно заинтересовать моддеров, - светодиоды, окрашивающие вентиляторы в синий цвет. Естественно, на такой скорости купер издает сильный шум - 35 дБ! Далеко не каждый сможет спать при таком шуме, исходящем из системника. Каждый из этих двух вентиляторов отдельно может подключаться как к разъемам Molex, так и к стандартному трехпиновому коннектору на материнской плате. При монтаже этого девайса придется демонтировать стандартное крепление для кулеров и установить разработанное инженерами Igloo, далеко не самое удобное. Но, как оказалось, со своей главной задачей, то есть с эффективным охлаждением, купер справляется на ура. Температура в простое колеблется примерно на уровне 31-32 градусов, а после подачи 100% нагрузки на процессор и такого разогрева в течении 30-ти минут температура повышается всего до 46-ти градусов. Стоит отметить, что тестирование проводилось на стенде открытого типа, без корпуса. При закрытом стенде в корпусе температура повысится на три-пять градусов.

Купер получился очень хороший и эффективный, но с одним существенным недостатком... Ну почему нельзя было разработать для этой модели регулятор скорости вращения вентиляторов в зависимости от нагрузки?! Есть же в ассортименте компании регулируемые по скорости кулеры...

Технические характеристики:
Совместимость: Socket 478, 603, 604
Материал радиатора: алюминий
Количество вентиляторов, шт: 2
Размер, мм: 100x98x108
Размеры вентиляторов, мм: 80x80x18
Уровень шума, дБ:
Скорость вращения вентиляторов, об/мин: 3200
Воздушный поток, CFM: 45,91
Вес, г: 800

Тестовый стенд
Материнская плата: Asus P4P800 SE
Процессор: Intel Pentium 4 3,4 ГГц (Northwood)
Видеокарта: Asus N6600GT 128 Mb
Память: 2x512 Мб DDR400 Hyundai/Hynix Original
Размеры вентиляторов, мм: 80x80x18
Жесткий диск: Maxtor 6Y120P0
Блок питания: 420Вт PowerMan Pro

SkyWriter (to@skywriter.ru)

ПАЯЛЬНИК

RESET'НЕМ ПО-БЫСТРОМУ?

Мы решили немного потомить тебя (или наоборот - позволить расслабиться) и слегка сменили тему на время: неужели тебе не надоело читать про все эти пластиковые карты и системы защиты? Признайся честно, что всем, в том числе тебе, хочется чего-нибудь разумного, доброго, вечного.

К такому разумному, доброму и вечному можно было бы отнести починку утюга для мамы, но мы хакеры, а не домохозяйки, так? :) Починим лучше, например, твой домашний маршрутизатор на базе Linux'а.

Безусловно, у тебя возникнет вопрос, а зачем его, собственно, чинить, если он и так почти всегда прекрасно работает? Вот он, корень зла: "почти всегда" прекрасно работает, а когда не работает, то, скорее всего, просто жестко висит, и ты в очередной раз давишь кнопку Reset на несчастном системнике. И в то же время ты радуешься, что маршрутизатор стоит у тебя дома, а не где-нибудь в далекой лифтовой или на крыше. А если он все-таки живет на крыше, то каждый твой зимний ночной поход к компьютеру будет просто праздником.

Проблему очертили, теперь будем решать ее? Решается она не просто, а очень просто: достаточно найти кого-нибудь/что-нибудь, кто/что перезагружало бы твой маршрутизатор, если тот завис. Ежу понятно, правда? А определить, что он завис, можно заставив его периодически "пингать" то самое кого-нибудь/что-нибудь: не дождались "пинка" - через минут пять перезагрузили маршрутизатор.

Итак, что будет делать наш девайс (работящего негра, денно и нощно ресетящего наш рутер, отбрасываем сразу, OK?) :)?

■ Он (девайс, который, кстати, грамотная общественность именует watchdog'ом) должен следить за работоспособностью рутера, обнаруживать сбои (вне зависимости от их природы: программный сбой или аппаратное "подвисание") и осуществлять "холодный" перезапуск компьютера в случае сбоя.

Рис. 1. Функциональная схема "собаки"

■ Слежку за работоспособностью ПК и выработку сигнала о нормальной работе системы (дальше, чтобы не париться - OK) реализуем программно: в рутере запускаем watchdog-драйвер - Unix-процесс, постоянно следящий за исправностью системы. Конкретный набор выполняемых проверок зависит от конфигурации, периферии рутера и твоих личных предпочтений.

Пока условие "все работает нормально" выполняется, наш процесс (watchdog-драйвер) с некоторым периодом t выдает аппаратный сигнал OK на интерфейс watchdog'а.

Аппаратная часть watchdog'а вырабатывает сигнал Reset в случае если в течение периода T>t не поступил сигнал OK подтверждения нормальной работы системы. То есть вне зависимости от причин сбоя (аппаратное "зависание", программные ошибки) потерянная работоспособность маршрутизатора будет гарантированно восстановлена без вмешательства негра :-). И тут ты спросишь: а как же проверка файловой системы после перезагрузки? Наверняка будут ошибки, и все равно придется залезать на крышу с монитором и клавиатурой.

Ошибки будут, если твой маршрутизатор налажен неграмотно. А грамотно - это так, чтобы в рабочем режиме все разделы, находящиеся на жестком диске/IDE-flash'ке (что именно используешь ты, не знаю), были замонтированы в режиме read-only, в этом случае даже при нажатии кнопки Reset они не будут помечены как "грязные" ("dirty" в терминологии man'ов). Есть, правда, некоторые разделы, которые не могут существовать в режиме read-only: /var, /tmp. Для них отдельная песня: при загрузке создаешь файловую систему где-нибудь на ramdisk'е, монтируешь ее в /var, /tmp и т.п. и создаешь там нужное дерево каталогов. При перезагрузке ramdisk, конечно, потеряется, но... для кого это важно? :)

Так вернемся к нашим "баранам". Для передачи от watchdog-драйвера к watchdog-аппаратуре сигнала OK предлагаю использовать стандартный разъем для подключения внутреннего динамика IBM PC разъем громкоговорителя :), известный также как "спикер" (не путать с обитателем Госдумы :) и присутствующий на материнских платах всех типов.

Чем привлекателен этот разъем? Тем, что не нужно занимать никаких слотов в маршрутизаторе, не нужно дорабатывать каким-либо образом материнскую плату твоего маршрутизатора (все можно оставить как есть - на базе 486SX ;-). Вдобавок использование разъема speaker как интерфейса между драйвером и аппаратурой watchdog:

■ заметно упрощает схему watchdog-аппаратуры, ее подключение и обеспечение электропитания;

■ упрощает программу драйвера: сигнал OK легко намутить просто пропищав что-нибудь невнятное в PC-speaker :-).

Функционально watchdog представляет собой одновибратор (ga-ga, так это у нас называется ;-) и состоит из двух узлов (см. рис. 1):

❶. генератор;

❷. счетчик.

Частота генератора составляет около 5,5 +/- 0,5 КГц. Такой выбор позволяет получить на выходе делителя сигнал с периодом T около трех минут. Такая величина задержки необходима для того, чтобы watchdog дождался (а не устроил ребут) выполнения загрузки операционной системы, запуска watchdog-драйвера и получения первого сигнала OK от него. Если твой маршрутизатор грузится больше трех минут, то знай - пора слезать с 386-ой платформы, она немного устарела :).

После запуска программного обеспечения дрова watchdog'а вырабатывают сигнал OK с периодичностью t<T. Этот сигнал сбрасывает в ноль счетчик (см. такую большую микросхему на рис. 1 ;-). При отсутствии сигнала сброса в течение периода T счетчик переполняется, и watchdog вырабатывает сигнал Reset, производя перезагрузку системы.

Кстати, счетчик watchdog'а сбрасывается не только в случае истошного пищания динамиком, в нем появляется ноль:

■ в момент включения компьютера - при появлении питания на watchdog-аппаратуре;

■ при поступлении сигнала Disable выключения watchdog-аппаратуры.

Последнее предусмотрено в схеме для того, чтобы ты мог легко отключить watchdog на время отладки или ремонта своего монстра, так как тогда драйвер watchdog может оказаться незапущенным. Этот самый сигнал Disable вырабатывается кнопкой Turbo системного блока твоего маршрутизатора.

Для индикации работы watchdog-аппаратуры используется светодиод TurboLED, который обычно присутствует в кузове. Если его нет (как и кнопки Turbo), то все это добро легко заменяется внешним переключателем и диодом.

Теперь "ближе к телу", а именно, к реализации. Схема watchdog'а (см. рис. 2) реализована на распространенной специализированной часовой микросхеме К176ИЕ12, имеющей в своем составе два инвертора и два делителя на 32768 и 60.

Генератор реализован на двух инверторах микросхемы DD1 (выводы 12, 13, 14) и внешних элементах R8, C3 и C4, задающих частоту генерации. Частота генератора подобрана достаточно высокой для обеспечения стабильности генерации и облегчения запуска генератора.

Счетчик собран на делителях микросхемы DD1, включенных последовательно.

Для индикации работы watchdog'а будем использовать светодиод Turbo LED корпуса системного блока (или тот, который ты прикрутил в качестве замены), мигающий с частотой, взятой с выхода делителя на 32768.

Сигнал сброса компьютера вырабатывается при появлении "1" на выходе делителя на 60 (вывод 4). Формирование сигнала производится транзисторами VT3 и VT4. Использование двух транзисторов позволяет подключать разъем reset с произвольной полярностью. Сигнал Reset держится включенным в течение 1/120 T, затем происходит сброс счетчика и снятие сигнала Reset. Это реализовано на транзисторе VT2, выполняющем функции логического "И" с одним инвертированным входом.

На диодах VD1, VD2 и транзисторе VT1 собран трехвходовый логический элемент "ИЛИ". Отключение watchdog'а производится кнопкой Turbo корпуса системного блока, при этом происходит установка на входе сброса счетчика логической единицы (оказалось удивительно просто и разумно, правда? Сам поразился :).

Кстати, не забудь при установке "следящей собаки" и подключении ее к материнской плате рутера учесть полярность разъемов Speaker и TurboLED, а то дело может получить фатальный исход.

Если на твоей матери в разъеме Speaker отсутствует "земля", можно использовать контакт земли с разъема Reset. Для этого следует вместо транзистора VT4 установить перемычку между его коллектором и эмиттером. После такой переделки нужно особенно внимательно следить за полярностью подключения watchdog-аппаратуры к разъемам Speaker и Reset (!).

Использованные при конструировании этого чуда детали приведены в таблице.

Если ты соберешь все правильно (да еще и по схеме, да еще и с соблюдением номиналов деталей :), эта штука не потребует наладки.

Замечу, кстати, что среди прочего установка watchdog'а в маршрутизатор повышает дистанционную управляемость системы: admin (то есть ты :) получает средство удаленного выполнения жесткого сброса маршрутизатора - для этого достаточно остановить

№	Обозначение на схеме	Наименование	Кол.	Примечание
		Конденсаторы		
1	C1,C2,C5	КМ-6-25В-100нФ	3	=
2	C3	КМ-6-25В-220пФ	1	=
3	C4	КМ-6-25В-3нФ	1	=
		Микросхема		
4	DD1	К176ИЕ12	1	=
		Резисторы		
5	R1,R12	МЛТ 0,125Вт-180 Ом 5%	2	=
6	R2	МЛТ 0,125Вт-1 кОм 5%	1	=
7	R3,R4,R5	МЛТ 0,125Вт-100 кОм 5%	3	=
8	R6,R7,R9-R11	МЛТ 0,125Вт-20 кОм 5%	5	=
9	R8	МЛТ 0,125Вт-200 кОм 5%	1	=
10	R13	МЛТ 0,125Вт-51 Ом 5%	1	=
		Транзисторы		
11	VT1,VT2	КТ3107Б	2	КТ3107
12	VT3-VT5	КТ3102Е	3	КТ3102Г,Д
		Диоды		
13	VD1,VD2	КД522	2	=
		Разъемы		
14	XP2	4-х штырьковый	1	=

Номенклатура деталей

грова watchdog'а и через время T (3 минуты, помнишь?) рутер будет жестко ребутнут.

В довершение ко всему приведу изображение печатной платы "собаки" в увеличенном масштабе. Печатная плата выполнена односторонней, с контактными площадками увеличенного диаметра и широкими (в районе 0,5-0,7 мм) проводниками. Это, на мой взгляд, вполне позволяет воспроизвести ее в кустарных условиях. К контактам XS1 и XS2 следует припаять проводники, заканчивающиеся соответствующими разъемами, для подключения к материнской плате.

Все! Продукт готов. Надеюсь, этот пресловутый watchdog-драйвер ты сможешь написать и сам. Благо дело, на BASH'е или PERL'е всего пара строчек ;-).

Ну а если возникнут какие вопросы, пиши, буду рад помочь!

Удачи в сетевом администрировании!

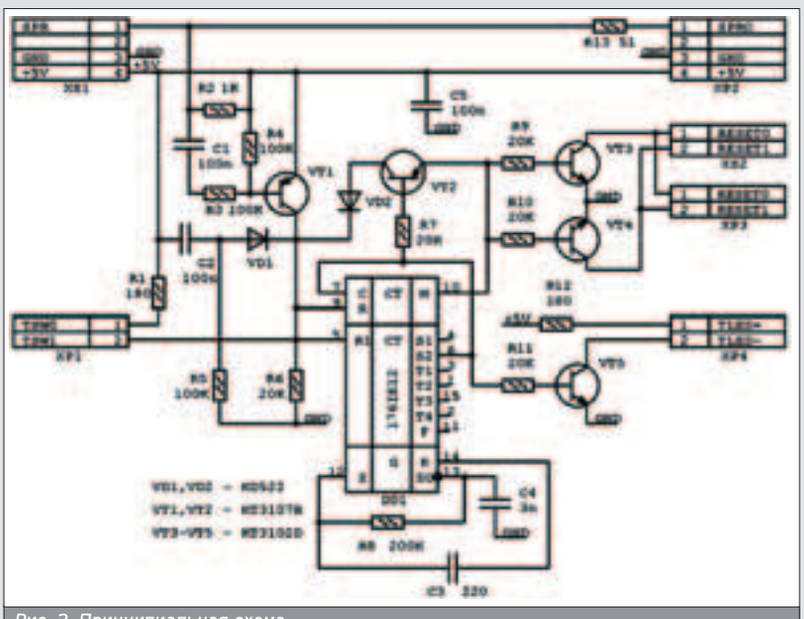


Рис. 2. Принципиальная схема

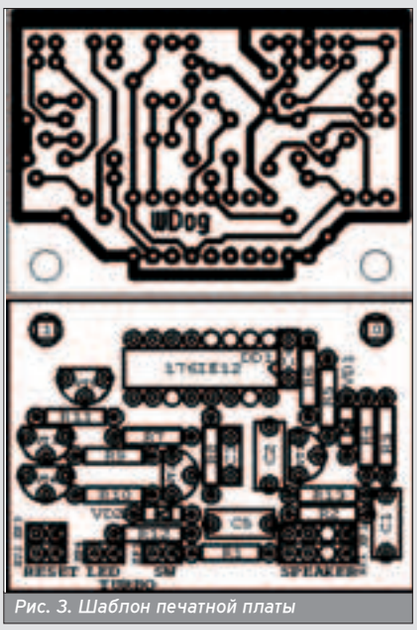


Рис. 3. Шаблон печатной платы

Dr.Klouniz (powered by клюквенная настойка)

E-МЫЛО

(spec@real.xakep.ru)

FROM: ГОБЛИН [BALACIR@MAIL.RU]
SUBJECT:

» Часто люблю лазить в чатах на MAIL.ру, нет ли у вас совета, как можно нелегальным путем стать модером или создателем.

ОТВЕТ:

Однако, вопрос! Человек так просто спрашивает меня, как ему стать Создателем. Да еще и нелегальным путем. Это уже ни в какие рамки не лезет. Ну ладно, слушай. Значит так. Чтобы стать Создателем, Творцом, Сверхчеловеком, убер зольдатом и Великим Кибермеханоидом Сущего, нужно: <CENSORED> в таблетках по 50мг + <CENSORED> в ампулах, лучше югославский. Таблетки в пасть, раствор по вене, завернуться в плед, выключить компьютер, одеть колпак-дурак на голову и залезть под стол. Лучше лицом вниз, а то можно подавиться рвотными массами. Удачи!

FROM: PARTIZAN [PARTIZAN_CHIK@MAIL.RU]
SUBJECT:

» Доброе время суток!
У меня вот какой вопрос есть к вам: слышал, что можно как-то на вашем сайте почтовый ящик зарегить, но вот беда - ссылки никак не мог найти.
Помогите пж...

ОТВЕТ:

Поздравляю сердечно! Право же, не каждый день нам пишут люди, возвратившиеся из летаргического сна, из комы или прилетевшие с Южного полюса от палатки Амундсена. Действительно, можно было сделать такое, но это было возможно годы и годы назад... С тех пор все изменилось. Люди стали летать в космос, хотя коммунистического будущего мы не достигли. Мобильная связь стала доступна всем и каждому. Многие люди пересели с модемов на выделенки, завершена программа "Геном человека", Шварценеггер стал губернатором Калифорнии, вышел Delphi2005, Аваланч разбил очки Горлу, ну и многое-многое другое. В общем, добро пожаловать в будущее!

FROM: VLADIMIR [ARKA@VI-MAIL.RU]
SUBJECT: БЛАГОДАРНОСТЬ

» Здравствуйте, Уважаемые!!!!!
От Ваших выпусков журнала (хакер, спецвыпуск хакер) заработал себе гемморой. Минусы: общение с женой, ребенком. Плюсы: море полезной инфы (мне 53 года), полная расслобуха(?) от реальной жизни и многое тп. и тд.
Так держать!
С уважением к молодым, Владимир.
P.S. С геммороем справлюсь сам :))

ОТВЕТ:

Вот так, Владимир! Вовремя ты. Сижу я с утра и плачу. Плачу и думаю, почему в глянцевые журналы все время пишут: "Ой спасибо, я читаю ваше издание, теперь я бросил пить, набрал 200 г. мышечной массы, потребляю не менее 0,1 мг. лейкопиена в день и снизил риск своей внезапной сердечной смерти на 0,05%. Спасибо вам, я теперь такой счастливый!" Или: "Мой муж читает вашу рубрику про секс, я впервые в жизни испытала оргазм, потеряла сознание, пришлось вызвать 03". Или еще круче: "Я подарил ротному свежий номер. Теперь не бьет меня по ночам и даже простил мне два обвала траншеи и взрыв на КПП". А нам такого не писали :(. Нам в основном: "Что за pdf? Почему Word'ом не открывается?", "Почему на почте мой журнал потеряли, я подписчик!", "ВзлАмай сайт, если ты меня уваажешь" ну и т.д. Так вот, расплакался и решил пойти почитать в спокойной обстановке. И вот допиваю я очередную бутылку, как вдруг - твое письмо. Респект. Кстати, у нас нет подарков за письма, но можем подарить свежий журнал. Заходи в редакцию, пароль - "Владимир54" :).

FROM: BEKBULAT [BEKBULAT@HOTMAIL.KZ]
SUBJECT:

» Здравствуйте, уважаемая редакция МС. Пишет Вам ваш постоянный читатель. Журнал просто супер, жаль только у нас трудно достать. Вы - моя последняя надежда!
А у меня ноутбук DELL Latitude v740 "японец" (кстати, про них у Вас ни слова). [ВЫРЕЗАНО]
Так вот мне знать где плюс где минус можно было запитаться от Compaq'ного блока. Может у кого-то есть блок питания от "японца" и на нем маркировка контактов, а может Вы и так знаете. Мне бы схемку.
Очень надеюсь на Вашу помощь, заранее благодарен мыло bekbulat@hotmail.kz

ОТВЕТ:

Так. Гоблин, это опять ты? Я же ясно сказал - завернуться в плед и залезть под стол. Компьютер с собой брать не надо. Пока тебя еще не очень вставило, срочно выключай его. А то начинается: уважаемая редакция "Мобильных компьютеров"! Как мне запитаться, дайте схему... ноутбуки, японцы... Да, торкает.

FROM: УСЗН [USZN_TS@TAGNET.RU]
SUBJECT: HELP!

Здравствуйте, уважаемая редакция журнала "Хакер Спец", разрешите поведать Вам душещипательную историю, суть которой будет ниже. Вот уже три с половиной года я - постоянный читатель вашего журнала. Сначала читал просто Хакер, а последние полтора года перешел на Хакер Спец. Хоть и бывает, что некоторые темы журнала специфичны и узконаправленны, как, например, тема вашего последнего номера - дизайн, но это нисколько меня не огорчает, поскольку пишете вы понятно, а главное - интересно, поэтому человеку, далекому от темы, рассмотренной в журнале, интересно будет узнать что-то новое, а человек, который уже занимался дизайном, пополнит и укрепит свои знания в этой области. Спасибо Вам за это.

Самое трудное и сложное в вашем, хотя уже могу сказать, в МОЕМ журнале - это...ждать. Ждать, когда же он придет. Считаешь дни, недели.

- Пришел? - вот вопрос, с которого начинается почти каждый мой рабочий день.

- Нет! - звучит как будто приговор из уст секретаря.

Но недавно заглядывает ко мне секретарша.

- Женя! Танцуй!! - сказала она.

- Наконец-то!!!! Ура!!! - но, взглянув на нее, я понял, что просто так за большие уши он мне не достанется... В общем, после зажигательного танца "Ламбада" под музыку "Аквариума" я прижал его к своему взмокшему беззащитному 120-килограммовому тельцу. Одним движением, которое уже стало безусловным рефлексом, освободил его из целлофанового плена, положил на стол и склонился над ним.

- Мое...моя прелесть, мое сокровище, - кажется, это где-то я уже слышал. Ну да ладно. Липкими трясущимися руками распечатал мешочек с диском и сунул его (диск) в сидюк, вожделея увидеть долгожданную инфу и т.д. и т.п.

И как вы думаете, что было дальше? ...а НИЧЕГО, да-да: Н И - Ч Е - Г О!!!

В общем, ничего хорошего я не увидел, кроме двух окошечек, которые Вам и направляю, чтобы и вы полюбовались на сие творение.

Объясните, пожалуйста, что могло пойти не так, в чем причина. На диске нет видимых повреждений и царапин. Хотелось бы получить от вас ответ на животрепещущий, надеюсь, не только для меня, но и для уважаемой редакции вопрос, т.к. журнал МОЙ стоит у МЕНЯ в Нижнем Тагиле аж целых 200 МОИХ рублей.

P.S.: На весь остаток дня погрузился в медитацию, в которой и пребываю по сей день.

ОТВЕТ:

Аналогично, коллега! Я тоже прослезился! С диском проблемы бывают, поскольку наша страна Велика и Могуча и никак в ней не получается найти завод, который не дает никакого процента брака. Трудно сказать, как помочь тебе. Могу, в принципе, оторвать от своего 77-килограммового тельца очередной Спец и подарить его тебе, только вот далеко ехать :(. Приезжай в Москву - подарим да еще и слово доброе скажем.

FROM: FOREWINTER@YANDEX.RU ON BEHALF OF SIEGFRIED [FOREWINTER@YANDEX.RU]
SUBJECT: ДОКТОР...

Здравствуйте, x-crew.

Намедни захотелось пообщаться с Доктором Добрянским. Я дважды писал ему на мыло, и два раза ко мне приходило сообщение об ошибке типа "нет коннекта с smtp.xakep.ru". Это безобразие! Немедленно сообщите мне его настоящие координаты! У меня еще есть вопрос. Народ, вы еще когда-нибудь собираетесь выпустить Спец по западлостроению или паяльнику? Киберманьяки ведь ждут :).

Всего хорошего любимой редакции! Sieg.

ОТВЕТ:

Мне вот интересно, зачем ты использовал smtp.xakep.ru и почему ожидал, что с ним будет коннект? Ты уверен, что такой SMTP вообще существует? Я вот сомневаюсь :). Используй другой. Про Спец по паяльнику подумаем, а западлостроение - вряд ли, мы отошли от этого.

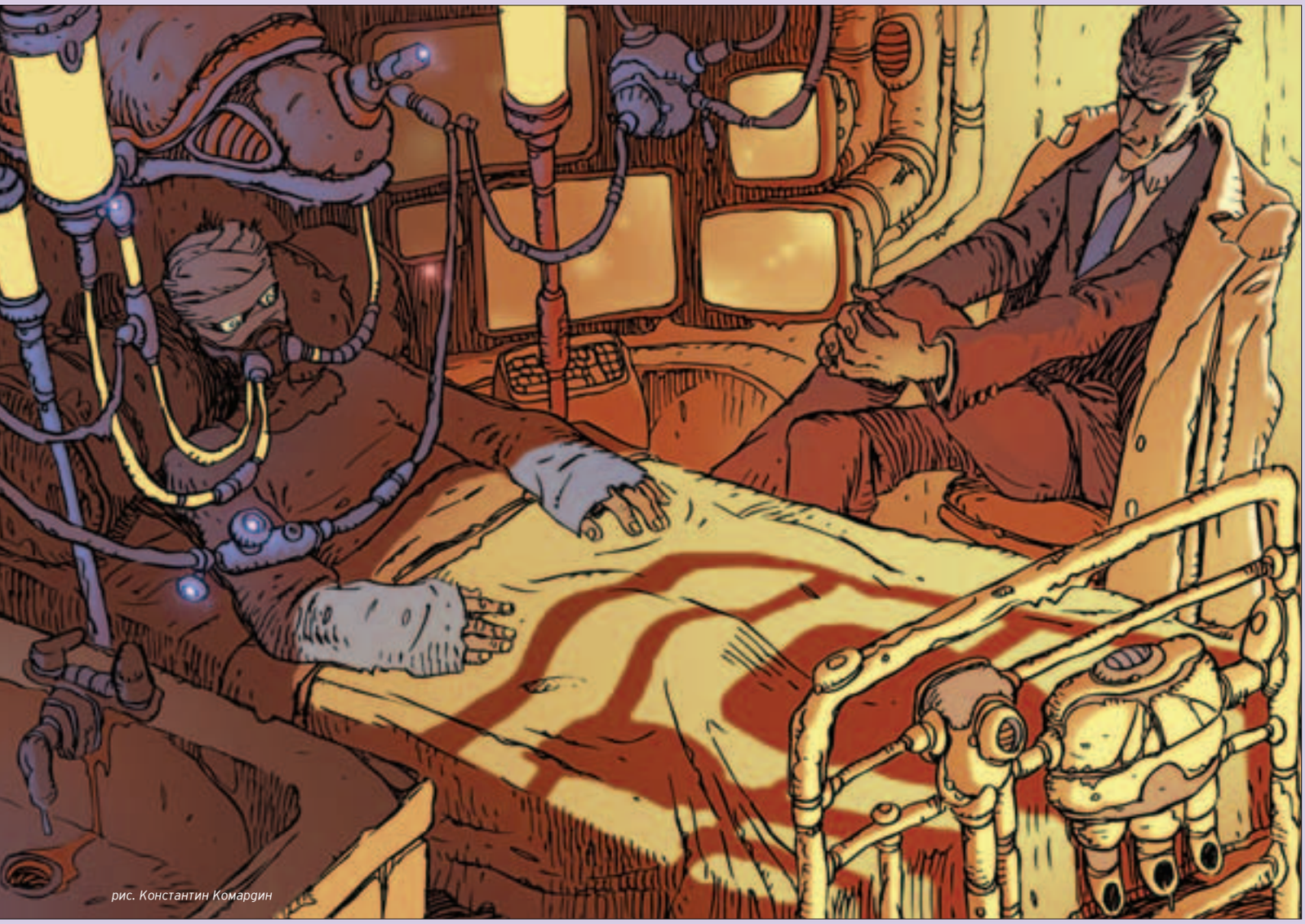


рис. Константин Комардин

Niro (niro@real.xakep.ru)

АТАКА НА ГИПОТАЛАМУС

(ПРАВДИВАЯ ИСТОРИЯ В ДВУХ ЧАСТЯХ)

ЧАСТЬ I

Миша хорошо запомнил тот комп, с которого все началось - 200-й "Пень" с шестнадцатью метрами оперативки, боронящий данные на маленьком винте в два гигабайта. Отец принес его в день рождения - в тот день Мишке исполнилось девятнадцать лет...

Детство прошло без всяких "Думов", "Квейков" и иже с ними: тогда их еще просто не было (а может, и были, да только кто ж знал?). Отрочество немного запомнилось программой на Бейсике, которую удалось написать на школьных курсах по информатике в одиннадцатом классе. Больше ничего примечательного не случалось до замечательной поры студенчества, когда судьба, опираясь на его же собственную смелость (или наглость?), зашвырнула его в самый большой ночной клуб города и поставила к диджейской стойке.

Как это произошло, он долго не мог понять сам: наврав с три короба о своих прошлых заслугах, которых попросту не было, он оказался за компьютерным пультом, на котором самым простым и понятным было включение в сеть, все остальное приходилось осваивать на ходу.

А потом он попал в гости к своему коллеге из клуба-конкурента и увидел у него... Лучше бы он этого не видел. Тот парень, Федор, уже тогда пытался приспособить к своему делу персоналку, и ему это удавалось, впрочем, с переменным успехом. Все, что он сделал, - это создал базу данных по хитам, где прописал служебную инфу и, самое главное, указал битовую скорость. Теперь по запросу он мог определить, какой хит наиболее близок по темпоритму к тому, что играет в данный момент, и выбрать из списка предлагаемых максимально удачный.

Миша стал все чаще бывать у Федора. Компьютер его интересовал с любой точки зрения. От музыки к играм, от игр к базам данных, от баз данных к инету... Многого он нахватался по верхам, читая книги для "чайников" и беседуя с Федором в паузах между управлением музыкой. Незнакомый мир приобретал все более четкие очертания, недоставало главного - собственного оружия для борьбы с компьютерной неграмотностью. Он начал тихую атаку на отца, тот поначалу отмахивался. Но постепенно на вопрос "А зачем тебе все это?" отец стал получать от Миши все более четкие ответы.

Однажды отец не выдержал:

- Конкретно что ты хочешь?

И Миша сказал. Через две недели отец купил ему "Пень" с предустановленной 98-й "виндой". Михаил понимал, что комп, скорее всего, бэушный (у отца точно не было денег на что-то покруче "двухсотого"), но и это его радовало.

Поудобнее устроившись на стуле, Миша протянул руку к кнопке Power и тихонько тронул ее, она мягко утопилась в корпусе и вернулась в прежнее положение. Зашумел вентилятор (Миша еще не знал такого слова, как "кулер"), застрочил внутри винчестер - чертовски красивое название. На мониторе загорелся маленький светлячок, раздалось тихое потрескивание размагничивателя. Миша от волнения потирал руки, как перед первыми турами в клубе, когда еще не до конца был уверен в себе и плохо знал содержимое дисков. На экране появились строки, из которых Миша понял только слова "Pentium-200 MMX", потом какие-то таблицы, голубые облачка с надписью Windows 98 и, наконец, то, что во всех самоучителях называлось рабочим столом - несколько значков на черном фоне, кнопка "Пуск" и часы. Миша машинально сверил время и осторожно дотронулся до "мышки". Стрелочка на экране шевельнулась.

И тут Миша понял, что перед ним - его собственный компьютер. Не надо просить никакого Федю о том, чтобы тот дал поработать пять минут, не надо выбирать время для того, чтобы друзья, у которых уже давно есть компы, оказались дома и не были заняты. Теперь можно делать все и в любое время. И его понесло.

Курсор метался по экрану как перепуганный, открывая и закрывая какие-то окна и меню, втыкаясь в значки и кнопки, "мышь" оказалась на удивление чувствительной, клавиатура - мягкой, все было чудесно. Минимальные познания, которыми обладал Миша к этому моменту, дали ему возможность вволю наиграться в карты, послушать музыку и пошариться в Проводнике. На этом знакомство с компом было закончено, он откинулся на спинку стула и удовлетворенно смотрел на картинку на Рабочем столе - огромный красивый аквариум (кто-то на предпродажной подготовке постарался!). "Сбылась мечта идиота!"

Зазвонил будильник. Михаил взглянул на часы - семь тридцать вечера. Пора было собираться на работу. Сумка с дисками уже стояла у дверей - всегда собранная, разложенная по группам фонотека, его гордость.

Всю жизнь мечтая оказаться там, где он сейчас работал, Михаил с ходу вошел в эту среду, как танк раздвигая конкурентов и поднимаясь в профессиональном плане на высокий уровень. Он сумел за короткий период пройти путь от того, что он называл "тамадой" (руководство клуба было уверено, что диск-жокей просто обязан бесконечно трепать языком, объявлять песни, поздравлять всякую братву с днями рождения - все эти заблуждения изрядно потрепали парню нервы), до диджея, до права коротко называться этими двумя английскими буквами. Он не стал придумывать себе прозвище, оставив все это Грувам, Гномам, Зайцам и Валдаям. Все звали его по имени "диджей Миша", и этого было достаточно.

Его напарник, с которым он делил тяжкий труд диджейства, Коля Трофимов, DJ Traffic, самостоятельно отошел на второй план, считая Мишу безусловным лидером в их дуэте. Он бесконечно консультировался с ним по поводу новой музыки, пытался разобраться в хитросплетении проводов и понять, почему из одной колонки вылетает один киловатт, а из двух - два ("Ведь она же тоже всего один киловатт!"). В общем, он устраивал Мишу, который ставил Коле твердую "четверку" за удержание толпы на танцполе.

Работали они обычно по два часа, потом менялись. За ночь выходило по две смены у каждого, что было в принципе не очень тяжело, особенно если молодежь оказывалась благодарной, готовой танцевать без конца и под любую музыку.

Сегодня у Миши было очень приподнятое настроение: подарок отца привел его в радостно-возбужденное состояние, работать хотелось как никогда и работать очень хорошо, душа требовала музыки и ритма не меньше, чем сто двадцать пять ударов в минуту. Выскочив из трамвая на остановке прямо напротив огромного здания ночного клуба Crazy House, он нырнул мимо знакомого охранника внутрь, забежал по лестнице на второй этаж и вошел в свое личное помещение - технический менеджер выделил ему пусть небольшую, но вполне приличную комнату для отдыха и хранения всяких дорогих необходимостей типа цифровых магнитофонов, супернаушников, подборки дисков, принадлежащих президенту клуба. На диванчике у окна уже валялся Коля, нацепивший на голову телефоны от "Панасоник", которыми страшно гордился, купив их на первую зарплату. Из них был слышен ритмичный шум: Трафик что-то там прослушивал, в такт подергивая ногами и руками и не замечая вошедшего.

Миша машинально сверил время и осторожно дотронулся до "мышки".

Миша поставил сумку на полку у входа, Трафик не обратил на это внимания - его взгляд блуждал где-то за окном. Сняв с вешалки свою фирменную футболку, привезенную друзьями из Сиэтла, он облачился в нее, надев на голову кепку (чтобы пот не заливал глаза), которая тоже входила в подарочный набор. Его друзья занимались импортом в Россию мясных полуфабрикатов, поэтому и на кепке, и на футболке были надписи "Cold Storage of Bellingham" (Беллингем - город-спутник Сиэтла). Перевел с английского эти слова Миша совсем недавно, когда ему вежливо намекнул на это его бывший школьный учитель, встретившийся случайно на улице. Оказалось, что Михаил вовсю рекламирует "Холодильные склады Беллингема", но деваться уже было некуда, к его одежде привыкли, да и он сам не хотел расставаться с такой унифромой. В этом было что-то вызывающее - ди-джей из холодильника! Да к тому же вряд ли кто-то еще, кроме учителя английского, сумеет перевести эти надписи.

- Слышь, Колян, - вдруг неожиданно для самого себя громко сказал Миша. - А мне отец комп подогнал!

Трафик молча показал большой палец на вытянутой правой руке, после чего в очередной раз сменил трек, отмотал первую минуту, как делал всегда, чтобы узнать, что там будет дальше, и только потом поднял глаза на Мишу.

- Какой?

- Чего ты орешь? - махнул на него Михаил. - "Пень" двухсотый. Мне понравился. Колян кивнул и снял телефоны. Уши у него были красные, мокрые. Трафик принялся растирать и массировать их.

- Двухсотый, говоришь? - переспросил он через минуту, все это время Миша терпеливо ждал продолжения разговора. - Сейчас скорости уже не те, двухсотый слабоват будет.

- Мне хватает! - возмутился Миша, который еще, конечно же, не понял, хватает ему или нет. - "Скорости не те!" Что ты понимаешь?

- Понимаю, - коротко ответил Коля. - У моей девчонки дома комп Целерон триста шестьдесят шестой. Летает - жуть! »

Миша закусил губу и разозлился - ничем нельзя похвастаться! Все этот Трофимов знает, все он видел!

- Летает... "А ты не летчик!" - знаешь такую песню?

- Да ну тебя, Миха! Что ты сразу закипел? Я ж не в обиду: двухсотый тоже хорошо, тем более для начинающих, - попытался оправдаться Трафик, поняв, что зацепил Мишу как не следует.

"Подумаешь, Целерон, - думал в это время Миша. - Все с чего-то начинают. Я вот с двухсотого "Пентиума", между прочим, Эм-Эм-Икс... Хотя хрен его знает, что это такое..."

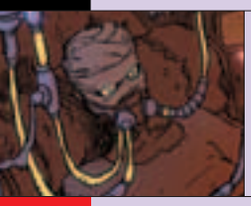
Тем временем Трафик подошел к большому стеллажу, взвалил на плечо коробку с дисками и вышел в дверь. Пора было подготовить все к началу рабочей ночи. Миша вытащил из своей бездонной сумки наушники (Sony, настоящие, никакой не Китай), нацепил их на шею, сунув штекер в карман брюк, и отправился следом.

Двери в зал были распахнуты. Внутри царил полумрак, лишний свет был отключен, только над диджейской стойкой горела тусклая лампа, выхватывающая из темноты кусок пульта. Где-то там уже копошился Трафик, слышался звук перекладывания дисков в коробке, потом лампа шевельнулась и нырнула под стол: Коля включал все, что должно быть включено.

Мишка подошел к тому, что они называли сценой - полутораметровое возвышение, окаймленное "гибким светом", германским "бегущим огнем" оранжевого цвета. Пара лестниц с обеих сторон вела наверх к стойке, выполненной полукругом и выкрашенной в черный цвет, и в темноте она просто исчезала. Из-под нее выбрался Трафик и принялся отряхивать колени: он тут вчера умудрился есть гамбургер и в итоге уронил его куда-то под ноги, а сегодня вляпался в его остатки.

- Что поставим для разминки? - оттирая кетчуп с колен, спросил, не поднимая головы, Коля. - Или не будем напрягать уши?

Миша улыбнулся - в его сумке лежал диск, который ему привезли два дня назад из Германии. Если он сейчас воткнет его просто так, по монитору, даже без базовых колонок, здесь через минуту будет весь персонал клуба. У "Скутера" вышел новый альбом No Time To Chill всего четыре дня назад, а знакомый пилот из "Транс-Аэро" привез ему эту новость вместе с диском. Дома Миша просто тащился от How much is the fish?, его так и подмывало поставить ее еще вчера, но из-за общего напряжения и усталости приберег хит до лучших времен. Ему в голову не могло прийти более прикольное название для хита - "Почем рыбка?" Миша твердил про себя и утром и вечером: "How much is the fish". Фраза была очень удачной, АБСОЛЮТНО НАВЯЗЧИВОЙ, хит - бесспорным. Он вспомнил девяносто третий год, когда из каждой машины орала Happy Nation. Судя по всему, эта судьба ожидает сейчас "Скутера", причем надолго.



Зуб прижал наушник к правому плечу, сделал погромче монитор и стал работать.

- Не будем напрягать уши, - ответил Миша, предвкушая ударный хит в середине своего тура. Трафик пожал плечами и продолжил свой туалет.

Время шло. Бармен, никогда не приходивший раньше чем за полчаса до начала работы, махнул им рукой, у него тоже была своя стойка - барная. Официанты уже тусовались в подсобке, начальство вновь появилось из ниоткуда, один из заместителей директора прошел мимо сцены нетвердой походкой, от него за версту разило перегаром. Ночная жизнь потихоньку набирала обороты.

Напротив сцены на втором этаже находилась аппаратная. Два огромных пульта: один для звукорежиссера, другой для лайт-мастера. Две огромных световых пушки, два ряда мощных усилителей Gemini. И где-то там сейчас был Леха (вообще-то Алексей Николаевич Кульков, но для всех просто Леха) - человек, на котором в клубе держалось все, от простой розетки до сканеров Galileo над головой у диджеев. Лучший в городе светорежиссер, он же гениальный электрик, он же просто хороший человек и любитель кофе Maxim с красной этикеткой.

Миша вытащил из кармана лазерную указку и махнул ей по второму этажу. Из темноты к горящей наверху настольной лампе вылезла ладонь Лехи с наколкой на тыльной стороне: какие-то непонятные змеи, нарисованные там еще по молодости и глупости. Ладонь махнула в сторону Миши и исчезла. Трафик, глядя на эту привычную процедуру, открыл левый лоток спарки (он всегда начинал с левого так же, как и Миша), вложил туда сборник и включил. Потом потихоньку вывел канал на примерно треть мощности.

Как всегда, в это мгновение все, кто был в клубе, на секунду подняли головы и прислушались. Сама работа диджея незаметна, но всякий раз, когда в зале появлялась новая мелодия, все знали, что где-то там, в темноте сцены кто-то включил диск. Леха мигнул фонарями рампы, крутанул зеркала сканеров: по танцполу метнулись яркие разноцветные звезды. Справа от стойки пыхнула струйкой дым-машина. Миша щелкнул вентилятором - клубы заструились на танцпол и постепенно растворились там. Зуб вспомнил те времена, когда вентилятора не было: сквозняк тянул дым прямо на стойку, и уже часа через два у напарников было ощущение, что они наелись известки, и хотя все убеждали их, что компоненты дыма абсолютно безвредны, вентилятор все-таки пришлось купить, так как диджеи наотрез отказались работать в такой обстановке.

- Чей тур на открытие? - спросил Трафик, втайне надеясь, что Миша возьмет на себя начало: он еще не все до конца прослушал из своих новых дисков. - Вчера ты был первый, значит, сегодня...

- Я начну, - услышал он в ответ. - Ты можешь пока отдыхать.

Коля ретировался в подсобку. Миша оглядел рабочее место, немного отодвинул монитор, стоящий с левого бока на высоте стойки (они долго с Трафиком спорили, с какой стороны должен "дуть" монитор: Миша привык подпирать "уши" правым плечом, а Коля левым, но в итоге Трафику пришлось переучиться и монитор встал слева), воткнул штекер и расправил провод, который очень любил перекручиваться. Проглядев разложенные диски, Миша несколько упорядочил их, в правый лоток вложил диск, обязанный стоять первым, и присел на вращающийся стул. Оставались считанные минуты до того, как первые посетители войдут в клуб.

Постепенно стулья у барной стойки заполнилась людьми, молодежь потягивала джин-тоник из высоких хайболов, пытаясь при этом красиво держать сигареты. Словно нехотя вспыхнули неоновые лампы по периметру зала и по второму этажу, тут же стали желтыми глаза и зубы и (о, боже, сколько раз напоминать!) засеребрилась перхоть на плечах тех, кто имел неосторожность без Head&Shoulders надеть черные футболки.

Некоторые из посетителей периодически оглядывались в сторону сцены, словно ожидая, когда же начнется то, ради чего они пришли сюда. И Миша начал - со своей любимой вещи. У каждого диджея есть такая, для души: она является словно сигналом к началу главного действа. У Миши это был регги Stop That Train (кто слышал, тот поймет). Он вообще полюбил регги, открыв этот стиль для себя совсем недавно. Вот и сейчас с первых аккордов у него поднялось настроение до невообразимых высот, танцоры вышли разминаться на танцпол, звезды полетели над залом - тур начался.

Зуб прижал наушник к правому плечу, сделал погромче монитор и стал работать. В лотки влетали диски, пальцы стремительно порхали над пультом: пич-бэндом разогнать или притормозить, пич-контролом зафиксировать, опять band, еще раз control, pause, play (благо кнопка большая, ярко-синяя, не промажешь), на таймере истекает последняя минута предыдущего трека, а уже начал играть следующий, кроссфейдер медленно скользит из левого положения в правое (а в мониторе все гладко, ровно, словно трек и не думает заканчиваться), и вот уже следующий хит лупит из шестикиловатных колонок. И так два часа. Непрерывный звук, непрерывный бит, Зуб перестал притоптывать ногой уже около полугода назад: практически невозможно два часа шлепать по полу, да и не поможет, если чувства ритма нет.

Постепенно танцпол заполнился трясущимися телами, дым-машина, выплевывая из себя завесу порциями, постепенно скрыла ноги молодых людей, время от времени наползая им на головы. Миша втянулся по полной программе - музыка была налажена, диск "Скутера" ожидал своей минуты, судя по часам, впереди еще было довольно много времени. Неожиданно он обратил внимание, что кто-то из персонала машет ему руками снизу, из-под сцены. Миша узнал старшего смены охраны, Дмитрия. Охранник протянул руку вверх, стараясь наклонить диджея как можно ближе к себе: перекричать грохот колонок было очень сложно.

- Президент здесь! - проорал он прямо в ухо Мишке. - Прошел в офис!

Миша кивнул, давая понять, что все услышал.

- Немного принял на грудь! - продолжал Дмитрий. - Сто процентов выйдет в зал! Ты уж не подкачай!

Зуб опять кивнул. Все, что от него требовалось в такие ночи, - не расслабиться, провести party на самом высоком уровне, чтобы хозяин остался доволен. Конечно, все остальные службы этого огромного спрута, каким был клуб, тоже напрягаются, чтобы произвести впечатление на президента (и ведь какой громкий титул себе придумал - "Президент"!). От этого многое зависит: будет ли вовремя зарплата, не полетят ли чьи-нибудь головы, выделят ли деньги на ремонт аппаратуры, разрешат или нет проводить своих друзей на концерты залетных "звезд", - в общем, зависит практически все. И Миша, будучи одним из составляющих клубного механизма, старался так же, как и все.

Он вернулся за стойку, втихомолку порадовался тому, как точно рассчитал время разговора с охранником и вновь вошел в привычный ритм,

периодически поглядывая на дверной проем, ведущий к офисным помещениям: оттуда в любую секунду могла появиться худощавая высокая фигура в строгом костюме - Его Величество Президент.

Ресторан при клубе быстро вылетел в трубу, оставив после себя наследство в виде нескольких столиков в одном из углов зала - для особо проголодавшихся гостей. А вот дискотека удержалась на достойном уровне, в чем была и заслуга Миши. Вообще, бригада, отвечающая за свет и звук, подобралась очень профессиональная и дружная, алкоголем не злоупотреблявшая, что в условиях клубной жизни было просто невероятным. Поскольку клуб был открыт на базе бывшего дома культуры, просуществовавшего до этого не один десяток лет, можно было представить, что творится со всеми коммуникациями в этом здании, не испытавшем прелести капитального ремонта ни разу с момента зачатия. Просто необходимы были люди, которые сумели бы поднять все это запущенное хозяйство на должный уровень за максимально сжатые сроки, и им это удалось. Хозяин понимал это, всячески поощрял техническую бригаду и держался за нее обеими руками (периодически придушивая).

В двери показался Валера - личный телохранитель Хозяина. Неброского вида, худощавый. И что в нем такого от телохранителя? Сразу и не скажешь, что у него за поясом "пушка", а дома на стене висит несколько дипломов по единоборствам. В общем, первое впечатление обманчиво.

Валера оглядел с порога зал, словно пытаясь взглядом раздвинуть дым, увидел рядом с дверью стул, присел. Следом за телохранителем появился и президент, судя по выражению лица, состояние его было близко к нирване. Он положил руку на плечо Валере и стал осматривать зал, притоптывая ногой, после чего что-то крикнул в ухо охраннику, наклонившись почти к самой голове Валеры. Тот кивнул, не отрывая глаз от танцующей толпы, вытащил из кармана кожаного жилета рацию, ответил на вызов, встал и предложил президенту пройти на выход. Хозяин устало кивнул, привалился к Валере и пошел рядом с ним.

Миша созрел для How Much Is The Fish. Он очень удачно подвел к этому хиту темпо-ритм, плавно опустил указательный палец на кроссфейдер, выдержал небольшую паузу и резко перевел его в нужную позицию. И зал просто завизжал от восторга! Волна энергетики захлестнула и Зуба, он почувствовал небывалый прилив эмоций, захотелось нырнуть в дым и цветные полосы и колбаситься вместе с тусовкой на танцполе...

Transforming the tunes!..

И тут в зале произошло какое-то движение, хорошо видное только с одной точки - от Мишиного пульта. Из-за дальнего столика, метрах в тридцати от сцены, встали два человека в плащах. Дымовая завеса словно специально разошлась на несколько мгновений, чтобы Зуб успел увидеть этих двоих.

We need your support!..

Почему они оказались в плащах, не смог потом ответить ни один человек - они словно возникли в зале не входя через главный вход. Просто встали из-за стола, и Зуб понял, что у них под плащами.

If you've got the breath back...

Валера продолжал вести Хозяина вдоль стены, подталкивая худеньким плечом молодежь, стоявшую с коктейльными стаканами в руках. Несмотря на невзрачный вид телохранителя, парни и девушки просто отлетали от него с возмущенными возгласами.

It's the first page of the second chapter!

Те двое, несомненно, продвигались в сторону Хозяина и с не меньшим, чем у Валеры, упорством. Пара молодых парней, упавших в угаре танца на пол от мощных толчков, даже не пыталась подняться: что-то во всем облике тех людей в плащах просто пригвоздило их к полу, они смотрели им в спины, боясь пошевелиться и не слыша музыки.

I want you back for the rhythm attack, coming down on the floor like a maniac... Две пары пробирающихся к выходу людей разделяло примерно двадцать-двадцать пять метров. И тут Миша пожалел, что у него нет микрофона.

I want you back, so clean up the dish by the way, How much is the fish?!!

Толпа безумствовала. Хит захватил их, тела изгибались в такт мощному биению хаус-пульса Бакстера и его команды. А "Титаник" приближался к своему айсбергу.

We're breaking the rules!

Зуб забыл о том, что нужно подбирать следующий хит. Надо было что-то делать. Судя по всему, он был единственным, кто видел сейчас всю картину в зале и предвидел столкновение, которое могло произойти в ближайшие несколько секунд. Адреналин ринулся в кровь, ноги стали ватными...

Ignore the machine...

В служебной двери показался Трафик: его привлекла незнакомая музыка. Он глядел снизу вверх на Мишу и махал ему рукой - классный трек приберег на сегодня! Зуб смотрел на него непонимающим взглядом и чувствовал, как рука тянется к ползунку уровня правого лотка, где сейчас играл "Скутер".

You won't ever stop this!..

Один из мужчин на секунду раньше другого сунул руку под плащ и вытащил пистолет. Оружие показалось в руке и у напарника. И тогда Миша решился.

Он резко уменьшил громкость, убрав ползунок практически до нуля, что вызвало сильный акустический шок у всех в зале - тишина ринулась в уши всем и каждому. Наверху в рубке Леха пролил кофе себе на шорты и по пояс высунулся вперед, пытаясь разглядеть диджея.

- Валера, справа! - что есть сил заорал Миша, перегибаясь через стойку вперед, словно это могло усилить его голос. В зале все автоматически посмотрели на диджея, подняв глаза на сцену. И только Валера

You won't ever stop this!..

среагировал профессионально - он резко толкнул Хозяина вперед, выпихнув его из сектора обстрела (тот покатился, как мячик, по пути зацепив пару стульев и официантку, уронившую на него поднос с грязными стаканами). Звон бьющегося стекла смешался со звуком первого выстрела.

Тут закричали все, кто только мог, и толпа рванула к дверям, на ходу сшибая друг друга и вдавливая тех, кто был пьян или менее поворотлив, в стены и пол. Паника охватила всех. Но киллеры продолжали выполнять свою задачу, стрельба шла с двух рук непрерывно. Валера, встав за одну из черных колонн, поддерживающих балкон второго этажа, даже не пытался выйти, а только взглядом зрителя мог смотреть на Хозяина, уползающего на корточках под столами в сторону выхода.

Миша невольно стал главным наблюдателем происходящего: он видел, как пули разрывали в клочья сначала стену, вдоль которой шел Президент, потом колонну, за которой скрылся Валера. Спустя несколько секунд стрельба перенеслась в тот угол зала, где полз Хозяин. В сторону отлетел парень с простреленной ногой - молча, не издав ни звука.

В зале неожиданно зажегся общий свет - Леха наверху прополз к щиту и врубил иллюминацию (вот только зачем?). Вентилятор постепен-

но разгонял дым. Хозяин сумел добраться до выхода, поднялся на ноги, мгновенно протрезвев, и рванул в проход, расталкивая локтями мечущуюся в панике толпу. Несколько пуль свистнуло у него над головой - киллеры выстрелили в открытую дверь наугад, еще один человек схватился за плечо и сполз по стене на пол.

Люди в плащах переглянулись, молча приняли какое-то решение и быстрым шагом направились к дверям. Перед ними все расступались, боясь пошевелиться. Миша, покрывшийся липким потом, посмотрел туда, где стоял Трафик: в дверях никого не было. Зуб вздохнул и облизнул высохшие губы. Пальцы нервно постукивали по стойке, крылья носа раздувались в такт дыханию, Миша был напуган как никогда в жизни. После такой ночи впору увольняться...

И тут один из киллеров обернулся в дверях и встретился взглядом с диджеем. Зуба словно током ударило: он и с тридцати метров почувствовал на себе пронзительный взгляд убийцы. Через секунду пистолет в руке киллера прыгнул на уровень глаз и прогремел выстрел. Девятимиллиметровый молот ударил Мишу куда-то в голову, отбросив на декоративную стенку за спиной. Проломив ее своим телом, Зуб повалился в переплетения опор, поддерживающих сцену, шнур от наушников со звуком, который издает тетива, лопнул, оставив штекер в гнезде. Сверху упали остатки стенки, полностью завалив Мишу и усыпав его щепками.

Коля, словно играя в прятки, высунул нос из служебного коридора и увидел разрушенную стенку у диджейской стойки.

- Миша... - тихо позвал он. - Ты где?

Парни и девушки, не успевшие выскочить на улицу, медленно выдвигались на танцпол из темных углов и молча смотрели туда, где сейчас умирал Зуб. Трафик взлетел на сцену, минуя лестницы, и начал руками разбрасывать завал из обломков над телом Миши. Когда прибыла милиция, Миша Зубков уже был в реанимации...

Аппарат зашипел, грудь сына стала ритмично подниматься и опускаться под простыней.

* * * * *

- ...Понимаете ли, уважаемый, я хочу сказать, что ряд странностей имеет место, - витиевато произнес доктор. - Странностей, не более того. Я не хочу сказать о чем-то сверхъестественном, "Истина где-то рядом..." и все такое. Я наблюдаю его с первого дня. Он всегда был тяжелым пациентом, и когда его привезли с простреленной головой, и потом, после операции, "тяжелым" в смысле "трудным", знаете, как говорили раньше - "трудный ребенок". С ним достаточно сложно общаться, он молчалив, все время проводит за компьютером, который вы ему принесли в палату, а ведь я был против! Рановато было нагружать мозг, ох как рановато!

Мишин отец виновато опустил голову. Он действительно нарушил распоряжение лечащего врача насчет компьютера, но ничего не мог поделать: сын так просил его об этом!

- Странности эти не приносят мне дополнительных хлопот, - продолжал доктор. - Напротив, они только помогают. Одно только заживление ран чего стоит! За восемь дней - то, что у других людей занимает две-три недели, а то и больше! И, конечно же, мощные возможности организма по восстановлению утраченных функций...

Отец кивнул, прекрасно понимая, о чем идет речь. О случившемся они узнали от Алексея Кулькова, который знал их домашний телефон. Он срывающимся от волнения голосом сказал, что их сын находится в Центральной клинической больнице, что его ранили из пистолета, попросил приехать туда и бросил трубку: он просто не мог больше ничего произнести от волнения. Отец собрался в несколько секунд, вылетел на лестницу и помчался в гараж. К больнице он подруливал уже через несколько минут, по дороге не замечая ни светофоры, ни разметку и знаки.

Его провели к завотделением реанимации, где вежливо, но настойчиво усадили в кресло и налили полстакана коньяка. И только потом отцу рассказали, что у Миши такая проблема: у него открытый огнестрельный пулевой многооскольчатый перелом лобной и правой височной костей с повреждением вещества мозга.

- В настоящий момент ваш сын жив, - тихо говорил врач. - Не буду скрывать от вас, что вероятность смертельного исхода невероятно высока, вам надо быть готовым к самому худшему. В эту минуту идет операция, после которой ваш Михаил будет помещен к нам, и вот тогда уже начнется самое серьезное, начнется борьба за жизнь. Бороться будем мы, вы и он. Цель у нас с вами одна, но пути ее достижения разные. Я и мои коллеги будем делать все от нас зависящее при помощи абсолютного научного подхода. Вы же должны делать все зависящее от вас в моральном плане: он не должен ни в чем нуждаться. Ему даже в коматозном состоянии необходимо слышать ваш голос или голос матери, он должен чувствовать заботу о нем. Лично я уверен, что больные с разными степенями расстройств сознания все равно присутствуют здесь просто потеряв способность к синтезу реакций. Так что... Мужайтесь и делайте все, что необходимо.

Отец полез в карман за деньгами - доктор остановил его.

- Не буду говорить, что я чертовски бескорыстен, но... Пока еще просто не за что.

Сына принесли в реанимацию через три с половиной часа. Отец видел издалека (ближе не дали пройти), как его аккуратно переложили на кровать и подключили трубку, торчавшую у него изо рта, к аппарату искусственного дыхания. Аппарат зашипел, грудь сына стала ритмично подниматься и опускаться под простыней. Голова была тщательно забинтована и покрыта сетчатым бинтом, но и сквозь него были видны проступающие розовые пятна. Отец прикусил губу и опустился в кресло, любезно подставленное доктором. Больше всего на свете ему хотелось сейчас найти хозяина клуба и пустить ему пулю в лоб. Устроили, сволочи, разборки! Лучше бы его убили, этого президента, лучше бы он...

- Доктор, - позвал он одного из дежурантов. - Он ведь может прийти в себя в любую минуту...

Врач виновато улыбнулся:

- Я бы не стал так надеяться на вашем месте. Я был на операции. Слишком уж тяжелое ранение...

Но все было иначе. Его сын очнулся уже в середине следующего дня, обвел глазами ту часть палаты, которую смог увидеть не поворачивая головы, и глазами показал на трахеотомическую трубку, часть которой ощущал в горле. Дежурная медсестра искренне удивилась и поспешила уведомить доктора о пришедшем в сознание больном Зубкове. А еще через двое суток, когда Мишу сняли с искусственной вентиляции, разрешили разговаривать с родителями и слушать маленький приемничек, появился Ник.

Вначале Миша, увидев незнакомого мужчину в халате, накинутом на плечи, решил, что это кто-то из клуба пришел справиться о здоровье. Мужчина опустился рядом с кроватью Миши на стул, обтянутый белоснежной простыней (так, чтобы Миша мог видеть его не напрягаясь) и вежливо улыбнулся:

- Привет. Как дела?

- Жив, - тихо сказал Миша. - Вы из клуба?

- Нет, - ответил мужчина. - Зови меня Ник. Теперь я буду часто приходить к тебе. Мне бы хотелось, чтобы мы подружились и нашли общий язык.

- Ник... - словно пробуя на вкус, произнес Миша. - Что вам надо? Вы от тех, кто стрелял? - пронзила его страшная догадка. - Это так?

- Не бойся, Михаил, - спокойно ответил Ник. - К тебе не так просто попасть. Если ты не в курсе, тебя охраняет несколько человек, ты важный свидетель преступления. На входе в реанимацию постоянно дежурит двое бойцов ОМОНа. Так что...

- Но ведь вы прошли! - достаточно громко сказал Миша, настолько громко, чтобы суметь привлечь внимание дежурного персонала. Но пара сестер и санитарка словно не замечали происходящего в реанимационном зале. Дежурный доктор вошел в двери, оглядел все четыре кровати, на каждой из которых лежали пациенты, подошел к Мише с другой от Ника стороны, пощупал пульс, потрепал его по плечу и отошел к столу заполнять историю болезни. Он словно не замечал посетителя, сидящего в двух метрах от него.

Миша недоумевающе посмотрел сначала на врача, потом на Ника. Тот жестом остановил готовый раздаться вопрос:

- Он не видит меня. Но не так, как не видят что-то отсутствующее.Я вполне материален, ты можешь прикоснуться ко мне. В настоящий момент изменено его отношение к реальности: он максимально сконцентрирован на том, что является для него сейчас самым важным - его работа. Именно поэтому все то, что к этой работе не относится, для него просто не существует.

Миша молча смотрел собеседнику в глаза и ждал продолжения: явно не он был инициатором и ведущим главной линии в разговоре, Ник просто должен был еще добавить что-нибудь для ясности. Тем временем тот сцепил пальцы рук перед собой, похрустел суставами и произнес после непродолжительного раздумья:

- Довольно скоро тебя переведут в травматологическое отделение. Врачи будут обращать на некоторые странности в твоем выздоровлении, скажем, оно будет идти неестественно быстро. Ты сам не должен этому

удивляться - это часть нашего с тобой контакта. Самое главное - как только ты будешь помещен в палату, попроси отца принести тебе компьютер. Врачи будут против - убеди отца. Тебе это не повредит. Можешь делать на компьютере все, что хочешь, лишь бы он был рядом. А сейчас я должен уйти. Помни мою просьбу, постарайся выполнить ее.

Ник поднялся со стула, подошел поближе и положил ладонь на забинтованную голову Миши - туда, где после перевязки проступили бисеринки крови. Глаза Миши сами собой закрылись, его окутало светящееся облако, и он ринулся куда-то вверх...

Ник, убрав через пару минут руку, осторожно вышел из реанимации и бодрым шагом прошел мимо охраны, повесив халат в шкаф для посетителей. Омоновцы продолжали беседовать друг с другом, не обратив никакого внимания на прошедшего к выходу человека...

На шестой день Мише разрешили сидеть - недолго, но все-таки это было большое достижение. На восьмые сутки нейрохирург скрепя сердце снял швы, рана зажила. Рентгенолог хватался за голову: снимки Михаила Зубкова до операции и на десятые сутки после нее разительно отличались, костная мозоль была видна даже без специальной подсветки. Парня было решено переводить в отделении травматологии для долечивания последствий ранения: сохранялись такие явления, как постоянное головокружение, частые головные боли, периодические расстройства сознания в виде обмороков. И тут Миша начал атаку на отца, он требовал принести компьютер. И отец, несмотря на многочисленные запреты врачей, сдался, к концу второй недели нахождения сына в больнице он привез ему "Пентиум". Через пять суток у Михаила полностью исчезли головокружения и обмороки, медикаментозное лечение которых по всем канонам медицины занимало не один месяц.

Врачи были просто в шоке - пациент не вписывался ни в одну из известных схем выздоровления. Михаил Зубков стал объектом исследования со стороны нескольких докторов наук, занимавших в больнице высокие посты. А он сам словно не замечал, что все происходящее имеет, мягко говоря, сверхъестественную оболочку. Да и что можно было хотеть от практически здорового через три недели после ранения молодого парня, который был до чертиков занят компьютером, даже не обращал внимания на молоденьких медсестер, откровенно увлеченных местным "феноменом". За этот короткий период Михаил освоил все основные прикладные программы, прочитав от корки до корки всю необходимую литературу. Он в совершенстве знал почти все профессиональные хитрости Windows, без страха и сомнения залезал в реестр и системные файлы, досконально изучил офисный набор Microsoft и даже набросал маленькую базу данных по учету стационарных больных для заведующего отделением травматологии, облегчившей тому во много раз статистический учет. И все это происходило на фоне звучащего из колонок "транса": отец принес по просьбе сына его сумку с дисками, уцелевшую тогда под обломками стены, и тот бесконечно крутил свою коллекцию во время "учебы".

Все это время Ник не появлялся, да Миша и не помнил о нем. С тех пор как тот исчез из реанимации, оставив Зуба в сияющем сне, ни одно воспоминание не тревожило сознание Миши. Он был полностью погружен в свой мир, пытаясь объять необъятное. Пару раз он намекнул отцу на то, что по приходу домой хотел бы подключиться к интернету (стало быть, нужен модем, хоть какой, простенький, лишь бы работал!). Папа вздыхал, но все делал на благо сына, подписал его на пару журналов компьютерной тематики, прикупил стол с выдвигающейся панелью для клавиатуры, поставив вместо своего у окна, занял денег на модем, но без сына покупать его опасался, боясь нарваться на что-нибудь недостойное, на "левое железо", как называл это на своем языке сын.

Миша просыпался и засыпал с компьютером, пройдя за одну ночь "Диабло" и с ходу выбрав удачную стратегию игры в StarCraft, после чего играть стало неинтересно. DOS был выучен одновременно с HTML по двум учебникам, лежащим рядом. Услышав от Трафика незнакомое слово "Линукс", произнесенное с изрядной долей презрения к "окошкам", освоил Red Hat, причем справочник команд стал для него настольной книгой не более чем на пару дней, потом он к нему уже не обращался, альтернатива воцарилась на его компьютере всерьез и надолго, он не страдал от того, что играть в его любимые игры стало практически невозможно, наоборот, сама "Красная Шапочка" была настолько интересна и увлекательна, что с лихвой компенсировала отсутствие игрушек.

Очень скоро завотделением потерял представление о том, зачем в палате №9 лежит абсолютно здоровый Михаил Зубков, о ранении которого напоминает только бледно-розовый шрам в правой височной области. Никаких объяснений случившемуся не было, Миша просто сам не понимал, из какой бездны он благополучно выкарабкался в такие краткие сроки.

Он тем временем серьезно занялся программированием под Линуксом, в очередной раз сменив набор книжек на тумбочке рядом с кроватью. Мать с ужасом смотрела на выросшую стопку непонятных ей книг, которые отец перетаскал из больницы домой. Она не могла понять, как ее сын за такой небольшой промежуток времени да еще после такой травмы сумел впихнуть в себя невообразимое количество материала. И еще во время очередного посещения в приемные часы родители обратили внимание на одну особенную деталь, не характерную для их сына: в палате стояла необыкновенная тишина, нарушаемая только пощелкиванием "мыши" и клавиш клавиатуры. Музыка не играла, сумка с дисками была собрана и упакована для отправки домой. Мама в душе перекрестилась: судя по всему, сын не собирался возвращаться в клуб.

В один из дней, незадолго до выписки, Миша, как обычно, занимался программированием. Так, ничего особенного, изучал очередные наборы команд. За окном стояла теплая погода, характерная для начала осени, рядом с открытой форточкой колыхнулась штора, Миша оторвался от монитора и взглянул на открывшуюся дверь.

В проеме стоял Ник. Миша мгновенно вспомнил свой сон с полетами в сверкающих искрах. Как и прежде, на плечи был небрежно накинут халат, он взял за спинку стул, стоящий у двери, пододвинул его к кровати Михаила и взглянул в экран. Все происходило молча, на уровне полного взаимопонимания. Ник внимательно просмотрел участок кода, видимый на экране, прокрутил немного вверх, усмехнулся и подправил пару значений.

- Ну как идут дела? - осведомился гость, осматривая голову Миши и переводя взгляд на кипу учебников по администрированию и по программированию. Миша пожал плечами:

- Сами видите, все хорошо. Понятия не имею, почему не отпускают домой. Все время приходят, смотрят, выслушивают-выстукивают... Надоело. Если бы не комп - от скуки бы помер.

- Очень хорошо, - удовлетворенно сказал Ник. - Так и должно было быть... Знаешь, Михаил, то, что произошло с тобой в клубе, - это случайность. Незапланированная трагическая случайность. Но то, что происходит после этого злополучного выстрела - тщательно продуманная часть плана. Он (то есть этот самый план) был бы реализован и в том случае, если бы с тобой ничего не произошло, но чертовы киллеры не только внесли коррективы в наши планы, но и оказали большое положительное влияние на сам процесс. Кстати, если тебе интересно, по твоим подробным описаниям этих людей взяли на восьмые сутки. К сожалению, трудно выйти на заказчиков покушения, но это тебя и меня не должно интересовать. В конце концов, стреляли все-таки не в тебя... То есть и в тебя тоже...

Ник внимательно просмотрел участок кода, видимый на экране, прокрутил немного вверх, усмехнулся и подправил пару значений.

Дверь распахнулась - вошел заведующий отделением Владимир Петрович.

- Вот... Зашел попрощаться, - начал доктор. - Сегодня выписываешься. Зашел пожелать удачи...

Миша краем глаза отметил, как Ник разглядывает свои ногти, делая вид, что он как бы не присутствует при этом разговоре.

- Ты очень необычный больной... Точнее сказать, САМЫЙ необычный, - продолжил тем временем Владимир Петрович. - Я хотел бы, чтобы ты иногда напоминал о себе, хотелось бы узнать, как сложится твоя дальнейшая жизнь... Если хочешь, звони (он протянул свою визитную карточку), всегда буду рад с тобой пообщаться. Как-никак, ты самая большая моя удача. Учитывая, что подобных операций я за свою практику выполнил около трех сотен...

Ник вежливо смотрел в сторону, словно извиняясь перед Мишей за свое присутствие, и что-то шептал себе под нос абсолютно неслышно. Тем временем врач вышел из палаты, напоследок потрепав Мишу по плечу. Ник встал, прошелся к тому месту, где стоял доктор, и выглянул в окно.

- Вон идет твой отец, - кивнул он в сторону улицы. - Он накопил денег на модем. Мой тебе совет, купи внешний, не лезь внутрь компа. Осваивай Сеть: мы скоро встретимся вновь, ты должен будешь многому научиться и многое уметь. Твоя жизнь наполнилась новыми целями и ощущениями, следуй им. Погрузись в интернет, найди там друзей. Я приду, когда ты будешь готов.

И он напоследок, проходя мимо Миши, вновь положил ему руку на лоб, рядом с рубцом. Вновь вспыхнули искры, засверкали звездочки и облака, но на этот раз Миша не заснул, а наоборот, почувствовал небывалый прилив сил и увидел все ошибки в коде, набранном за последние »

двое суток. К тому времени Ника уже в палате не было, в дверях показался отец с большими пустыми сумками. Пора была собираться домой...

Дома уже ждал Трафик, он все время интересовался здоровьем друга, периодически навещал его в больнице, пытался снабжать новостями клубной жизни и музыки. Коля сидел в кресле у телевизора и сразу подскочил, когда услышал, как Миша вместе с отцом вошли в дверь.

- Привет! - радостно закричал Трафик. - Ты не выглядишь больным! Давайте помогу! - присоединился он к Мишиному отцу, подхватив большую сумку, в которую тот сумел засунуть монитор. Миша прошел в зал и увидел у окна большой компьютерный стол, на который отец уже положил клавиатуру и "мышку".

- Здорово! - выдохнул он. - Спасибо, па, - поблагодарил он и сразу полез под стол, чтобы установить компьютер, потом подошел к отцу, пожал ему руку и спросил:

- Ты обещал модем - как, получится?

Папа кивнул.

- Хоть сейчас. Я сам боялся покупать: вдруг ты меня потом ругать будешь на чем свет стоит! Но сам понимаешь, что-нибудь крутое вряд ли получится, доктор правду сказал в первый день, он действительно не очень бескорыстный.

Миша улыбнулся, вспомнив Владимира Петровича.

- Это само собой, папа. Я думаю, что его работа стоит того, что ты ему дал, а может, и больше. А если ты и в самом деле можешь прямо сейчас, зачем откладывать?

Через пару часов нырнув (на этот раз с модемом) вновь под стол, он усмехнулся. Судя по всему, он много времени будет проводить здесь в будущем, возиться с железом ему представлялось крайне интересным. Корпус был извлечен на стул, винтики откручены, крышка снята. На внешний модем им не хватило денег...

Миша потер правое запястье и, вспомнив главу из самоучителя по работе в интернете, набрал в адресной строке www.rambler.ru.

Впервые Миша увидел свой компьютер изнутри. За три месяца, проведенных в больнице, он так и не решился этого сделать. Сегодня он заглянул во внутренности своего компа и поразился тому количеству пыли, которая скопилась там в углах и на платах. Первым делом Миша вытащил пылесос и аккуратно прибрал слои пыли, которые свободно лежали, остальное вытер еле-еле влажной тряпкой и просушил феном. После таких процедур он пристально взглянул на "железо". Ничего особенного - все, что он и ожидал увидеть. А это что за фигня?

Миша удивился тому, что увидел: в один из PCI-разъемов рядом с видео-картой был воткнут девайс, у которого не было порта. Размером он был такой же, как и "видюха", тот же темно-зеленый материал (Миша никак не мог запомнить его название), переплетение мостиков, какие-то радиодетали, несколько светодиодов, в середине что-то большое, напоминающее по размерам процессор. Миша потрогал - плата была теплая, несмотря на то что последние два с половиной часа компьютер был выключен и все остальные устройства в нем были холодными.

Миша даже забыл, что комп он разобрал для того чтобы вставить туда модем. Ему в голову не могло прийти ни одно объяснение этой детали. Он присел рядом с компьютером на пол и уставился на девайс, скорчив на лице гримасу полного непонимания. Спросить в настоящий момент было не у кого. Решив узнать, что это такое, в самое ближайшее время, Миша разобрался с новым приобретением, модем встал как влитой, крышка вернулась на место.

При подключении модем зашумел и засвистел: Миша еще не знал, как сделать тише динамик, пришлось выслушивать все это пение до тех пор, пока в комнату не вошел отец, привлеченный посторонним незнакомым звуком.

- Все путем? - поинтересовался отец, сложив газету под мышку.

- Ага, - кивнул Миша. - Сейчас что-то будет.

Папа придвинулся поближе и вместе с сыном стал смотреть в экран. И в этот момент у Миши ощутимо зачесалось правое запястье - там, где у Ника ("Ник..." - всплыло имя из глубины памяти) был браслет.

- Что думаешь найти? - поинтересовался отец.

Миша потер правое запястье и, вспомнив главу из самоучителя по работе в интернете, набрал в адресной строке www.rambler.ru. Отец заинтересованно смотрел, как на экране постепенно появляется страница одного из самых больших поисковиков в России. А в это время с Мишей творилось что-то непонятное: вместо того, что видел отец - шапку сайта, баннеры, новости - перед ним сам собой раскрывался исходник, он видел все ошибки разработчиков, все "дыры" в защите, все возможности взлома сайта. При этом Миша каким-то "десятым чувством" сумел определить адреса всех сайтов, спрятанных за баннерами, и вычислить, какие из них наименее защищены. Он не стал говорить об этом отцу, так как и сам не понимал, что происходит, как это можно описать словами. Все ссылки на странице были видны ему в виде IP-адресов, он представлял себе всю географию интернета по ним: он ЗНАЛ, где и что находится.

Тем временем отец в недоумении толкнул сына в плечо:

- Что-нибудь будет происходить? Время-то идет, сын.

Миша очнулся, заметил, что до сих пор трет кожу на правом запястье. И в ту же секунду он увидел обычный вид главной страницы "Рамблера". Полазив в Сети минут пятнадцать-двадцать, Миша и его родители остались довольны первым опытом общения с Сетью, мать с отцом ушли спать, а сын, отключившись от Всемирной паутины, прилег не раздеваясь на кровать и попытался найти разумное объяснение тому, что он увидел сегодня при открытии страницы поисковика. Объяснения не было. Да его и не могло быть, все это было из области нереального.

Ворочаясь на кровати, Миша изучал тени на потолке. Плохо представляя себе возможности, открывающиеся перед ним, он размышлял об уровне своих познаний. Неожиданно зазвонил телефон. Миша сел на кровати и уставился на аппарат. Звонок раздался вновь. Миша схватил трубку, боясь, что родители успели услышать.

- Да...

- Привет. Это Ник.

(В памяти всплыла фигура в плаще и с браслетом на руке - человек из сияющего сна).

- Привет.

- Ты видел?

- Да. Что это?

Пауза. Миша просто чувствовал, как Ник сидит в глубоком мягком кресле и, закинув ногу на ногу, разглаживает полу плаща.

- Теперь так будет всегда. Не бойся, ты не сошел с ума. Научись контролировать эту способность. Ныряя в Сеть, без нужды не старайся войти вглубь страниц, смотри поверх - это то, что люди называют "серфингом", скольжением.

- Зачем мне все это? - тихо спросил Миша, поглядывая на дверь. - Не волнуйся, родители не войдут, - произнес Ник, будто видя Мишины тревожные взгляды. - А зачем тебе все это - тема для отдельного разговора. Для него еще придет время...

- Сейчас! - чуть не закричал Миша. - Я не смогу так видеть исходники, знать, как сломать защиту...

- Это все откровенная чушь, - засмеялся на том конце провода Ник. - Подумаешь, он увидел теги на странице! Да любой в состоянии это сделать, выбрав нужный пункт меню в браузере. Твои возможности гораздо шире! Я даже не в состоянии оценить их все и объяснить в нескольких словах!

Ник замолчал, тяжело дыша в трубку. Наконец, он собрался с силами и продолжил, на этот раз довольно неприятным тоном.

- Не стану скрывать: все связанное с тобой происходит как-то незапланированно. Чего стоит только твое ранение - абсолютно непредусмотренный случай! Потом эти ребята в переулке... И то, что у вас не хватило денег на внешний модем. Я так не хотел, чтобы ты раньше времени нашел инициатор...

В голосе Ника проскакивало неподдельное сожаление. Что же это за плата такая - инициатор?

- Если хочешь, подключись сейчас. Логин "Fly", пароль "Visitor". Телефоны те же, что дал тебе провайдер.

- Fly? Муха? - переспросил Миша.

- Да. Я так захотел. Это теперь твой логин для Unlimited. При родителях не пользуйся этим доступом, плати деньги за настоящий. Знай, что пока ты свободен в своих поступках. И ты жив благодаря нам.

- Кому "нам"? - недоуменно спросил Миша.

- Тем, кто вставил инициатор в твой комп за день до твоей клинической смерти. Это мы не дали тебе умереть.

В трубке раздались короткие гудки - Ник закончил разговор. Миша опустил трубку себе на колени, забыв положить ее на на место...

ПРОДОЛЖЕНИЕ СЛЕДУЕТ...

LG
Life's Good

FLATRON™
freedom of mind

FLATRON F700P

Абсолютно плоский экран
Размер точки 0,24 мм
Частота развертки 95 кГц
Экранное разрешение 1600x1200
USB-интерфейс

Dina Victoria
(095) 688-61-17, 688-27-65
WWW.DVCOMP.RU

Москва: АБ-групп (095) 745-5175; Акситек (095) 784-7224; Банкос (095) 128-9022; ДЕЛ (095) 250-5536; Дилайн (095) 969-2222; Инкотрейд (095) 176-2873; ИНЭЛ (095) 742-6436; Карин (095) 956-1158; Компьютерный салон SMS (095) 956-1225; Компания КИТ (095) 777-6655; Никс (095) 974-3333; ОЛДИ (095) 105-0700; Регард (095) 912-4224; Сетевая Лаборатория (095) 784-6490; СКИД (095) 232-3324; Тринити Электроникс (095) 737-8046; Формоза (095) 234-2164; Ф-Центр (095) 472-6104; ЭЛСТ (095) 728-4060; Flake (095) 236-992; Force Computers (095) 775-6655; ISM (095) 718-4020; Meijin (095) 727-1222; NT Computer (095) 970-1930; R-Style Trading (095) 514-1414; USN Computers (095) 755-8202; ULTRA Computers (095) 729-5255; ЭЛЕКТОН (095) 956-3819; ПортКом (095)777-0210; **Архангельск:** Северная Корона (8182) 653-525; **Волгоград:** Техком (8612) 699-850; **Воронеж:** Рет (0732) 779-339; РИАН (0732) 512-412; Сани (0732) 54-00-00; **Иркутск:** Билайн (3952) 240-024; Комтек (3952) 258-338; **Краснодар:** Игрек (8612) 699-850; **Лабытнанги:** КЦ ЯМАЛ (34992) 51777; **Липецк:** Регард-тур (0742) 485-285; **Новосибирск:** Квеста (38322) 332-407; **Нижний Новгород:** Бюро-К (8312) 422-367; **Пермь:** Гаском (8612) 699-850; **Ростов-на-Дону:** Зенит-Компьютер (8632) 950-300; **Тюмень:** ИНЭКС-Техника (3452) 390-036.

ИТ–решения Samsung для бизнеса

Не секрет, что многие преуспевающие компании выбрали технику Samsung для построения внутренней информационной структуры. Продукты Samsung помогают добиваться успеха в бизнесе как глобальным корпорациям, так и небольшим фирмам. Революционные технологии, используемые в наших ноутбуках, печатных устройствах и мониторах, позволяют Samsung по праву называться ведущей ИТ–компанией.

Галерея Samsung: г. Москва, ул. Тверская, д. 9/17, стр. 1.
Информационный центр: 8–800–200–0–400. www.samsung.ru. Товар сертифицирован.

SAMSUNG