

ТЕХНИКА ОТЛАДКИ ПРОГРАММ БЕЗ ИСХОДНЫХ ТЕКСТОВ

ОСНОВНОЙ
ИНСТРУМЕНТАРИЙ ХАКЕРА

ВЗЛОМ ПРОГРАММ
С ЗАКРЫТЫМИ ГЛАЗАМИ

“ПРОТИВОУГОННЫЕ”
СИСТЕМЫ
СВОИМИ РУКАМИ

БОРЬБА С КРИТИЧЕСКИМИ
ОШИБКАМИ ПРИЛОЖЕНИЙ

ВНЕДРЕНИЕ И УДАЛЕНИЕ
ВИРУСНОГО КОДА
ИЗ РЕ-ФАЙЛОВ

ТЕСТИРОВАНИЕ
ПРОГРАММНОГО
ОБЕСПЕЧЕНИЯ



PRO
ПРОФЕССИОНАЛЬНОЕ
ПРОГРАММИРОВАНИЕ

Крис Касперски

**ТЕХНИКА
ОТЛАДКИ
ПРОГРАММ
БЕЗ ИСХОДНЫХ ТЕКСТОВ**

Санкт-Петербург

«БХВ-Петербург»

2005

УДК 681.3.06
ББК 32.973.26-018.1
К28

Касперски К.

К28 Техника отладки программ без исходных текстов. — СПб.: БХВ-Петербург, 2005. — 832 с.: ил.

ISBN 5-94157-229-8

Даны практические рекомендации по использованию популярных отладчиков, таких как NuMega SoftIce, Microsoft Visual Studio Debugger и Microsoft Kernel Debugger. Показано, как работают отладчики и как противостоять дизасемблированию программы. Описаны основные защитные механизмы коммерческих программ, а также способы восстановления и изменения алгоритма программы без исходных текстов. Большое внимание уделено внедрению и удалению кода из PE-файлов. Материал сопровождается практическими примерами.

Компакт-диск содержит исходные тексты приведенных листингов и полезные утилиты.

Для программистов

УДК 681.3.06
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. гл. редактора	<i>Игорь Шишигин</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Елена Кашлакова</i>
Компьютерная верстка	<i>Натальи Караваевой</i>
Корректор	<i>Наталья Першакова</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Игоря Цырульниковца</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 15.08.05.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 67,08.

Тираж 3000 экз. Заказ №

"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию № 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-229-8

© Касперски К., 2005
© Оформление, издательство "БХВ-Петербург", 2005

Оглавление

Предисловие	1
Об авторе.....	1
О чем и для кого эта книга.....	3
Введение	11
История хакерства.....	11
История происхождения термина "хакер".....	14
Психология хакера	16
Лаборатория искусственного интеллекта и PDP-1	20
Сеть.....	23
Си и UNIX.....	26
Конец хакеров шестидесятых	33
RSX-11M	36
Intel	37
Хаос.....	38
Бытовой компьютер восьмидесятых	40
Рождение современных хакеров, или снова Intel	41
Глава 1. Знакомство с отладочными инструментами	45
1.1. Как работает отладчик.....	48
Обработка исключений.....	50
1.2. Что нам понадобится.....	51
1.3. Особенности отладки в UNIX.....	53
PTrace — фундамент для GDB.....	56
PTrace и ее команды	58
Поддержка многопоточности в GDB.....	60
Краткое руководство по GDB.....	61
Трассировка системных функций	66
Интересные ссылки	67

1.4. Эмулирующие отладчики и эмуляторы.....	68
Минимальные системные требования.....	70
Выбирай эмулятор себе по руке!.....	71
1.5. Обзор эмуляторов.....	74
DOSBox.....	74
Bochs.....	76
Microsoft Virtual PC.....	77
VMware.....	79
Сводная таблица характеристик эмуляторов.....	80
Разные мелочи.....	81
1.6. Области применения эмуляторов.....	82
Пользователям.....	82
Администраторам.....	83
Разработчикам.....	84
Хакерам.....	87
Как настроить SoftIce под VMware.....	89
Экзотические эмуляторы.....	89
1.7. Кратко об эмуляции процессора.....	90
1.8. BoundsChecker.....	96
Быстрый старт.....	98
Подключение нестандартных DLL.....	101
Пункты меню.....	103
1.9. Хакерские инструменты под UNIX.....	107
Отладчики.....	107
Дизассемблеры.....	111
Шпионы.....	112
Шестнадцатеричные редакторы.....	114
Дамперы.....	115
Автоматизированные средства защиты.....	115
Глава 2. Защитные механизмы и их отладка.....	119
2.1. Классификация защит по роду секретного ключа.....	120
2.2. Создаем защиту и пытаемся ее сломать.....	123
2.3. От eхе до сгк.....	125
2.4. Знакомство с отладчиком.....	142
Бряк на оригинальный пароль.....	143
Прямой поиск введенного пароля в памяти.....	160
Бряк на функции ввода пароля.....	170
Бряк на сообщения.....	173
Механизм сообщений в Windows 9х.....	176
2.5. На сцене появляется IDA.....	177
2.6. Дизассемблер & отладчик в связке.....	209
Из языка IDA-Си.....	212

2.7. Дао регистрационных защит.....	216
Как узнать имя функции по ординалу	221
Как сделать исполняемые файлы меньше.....	248
Перехват <i>WM_GETTEXT</i>	249
2.8. Хеширование и его преодоление	251
2.9. Ограничение возможностей.....	267
2.10. Ограничение времени использования	286
2.11. Ограничение числа запусков	291
2.12. NagScreen.....	293
2.13. Ключевой файл.....	302

Глава 3. Противостояние отладке 315

3.1. Обзор способов затруднения анализа программ	317
3.2. Приемы против отладчиков реального режима.....	319
3.3. Приемы против отладчиков защищенного режима.....	335
3.4. Как противостоять трассировке	348
3.5. Как противостоять контрольным точкам останова.....	354
Несколько грязных хаків, или как не стоит защищать свои программы	362
Серединный вызов API-функций.....	363
Вызов API-функций через мертвую зону	382
Копирование API-функций целиком.....	385
Как обнаружить отладку средствами Windows.....	388
3.6. Антиотладочные приемы под UNIX.....	389
Паразитные файловые дескрипторы.....	390
Аргументы командной строки и окружение	391
Дерево процессов	392
Сигналы, дампы и исключения.....	392
Распознавание программных точек останова	393
Мы трассируем, нас трассируют.....	394
Прямой поиск отладчика в памяти	395
Измерение времени выполнения	396
3.7. Основы самомодификации	396
Проблемы обновления кода через Интернет	409
3.8. Неявный самоконтроль как средство создания неломаемых защит.....	411
Техника неявного контроля	413
Практическая реализация.....	415
Как это ломают?.....	425
3.9. Ментальная отладка и дизассемблирование	435
Маленькие хитрости.....	459
3.10. Ментальное ассемблирование	460
3.11. Краткое руководство по защите ПО.....	465
Джинн из бутылки, или недостатки решений из коробки.....	466

Защита от копирования, распространения серийного номера.....	466
Защита испытательным сроком.....	467
Защита от реконструкции алгоритма.....	467
Защита от модификации на диске и в памяти.....	469
Антидизассемблер.....	470
Антиотладка.....	470
Антимонитор.....	471
Антидамп.....	471
Как защищаться.....	473
Мысли о защитах.....	474
Противодействие изучению исходных текстов.....	474
Противодействие анализу бинарного кода.....	478
3.12. Как сделать свои программы надежнее?.....	481
Причины и последствия ошибок переполнения.....	482
Переход на другой язык.....	483
Использование кучи для создания массивов.....	484
Отказ от индикатора завершения.....	484
Обработка структурных исключений.....	485
Традиции и надежность.....	487
Как с ними борются?.....	488
Поиск уязвимых программ.....	489
Неудачный выбор приоритетов в Си.....	493
3.13. Тестирование программного обеспечения.....	495
Тестирование на микроуровне.....	497
Регистрация ошибок.....	498
Бета-тестирование.....	499
Вывод диагностической информации.....	502
Верификаторы кода языков Си/Си++.....	504
Демонстрация ошибок накопления.....	505
Глава 4. Примеры реальных взломов.....	509
4.1. Intel C++ 5.0.1 compiler.....	510
4.2. Intel Fortran 4.5.....	518
4.3. Intel C++ 7.0 compiler.....	523
4.4. Record Now.....	532
4.5. Alcohol 120%.....	535
4.6. UniLink v1.03 от Юрия Харона.....	549
UniLink v1.03 от Юрия Харона II, или переходим от штурма к осаде.....	571
Entry Point и ее окружение.....	572
Передача управления по структурному исключению.....	574
Внутри обработчика.....	581
Таинства stealth-импорта API-функций, или как устроена <i>HaronLoadLibrary</i>	586

Таинства stealth-импорта: как устроена <i>HaronGetProcAddress</i>	589
Таинства <i>IsDebuggerPresent</i>	599
Таинства загрузки USER32.DLL и ADVAAPI32.DLL.....	601
Конец таинств, или где тот trial, который expired.....	604
4.7. ARJ.....	613
4.8. AVPVE: разбор полетов.....	614
Формат файлов (общее).....	615
Формат файла языковой поддержки avp.lng.....	616
Формат файлов HLP.....	617
4.9. Bounds-Checker 5.....	620
4.10. CD-MAN EGA Version.....	621
4.11. F-PROT 2.19.....	622
4.12. FDR 2.1.....	627
4.13. HEXEDIT.EXE Version 1.5.....	629
4.14. SGWW password protection WhiteEagle.....	630
4.15. SOURCER 5.10.....	631
4.16. Emulated Solar CPU.....	632
4.17. POCSAG 32.....	635

Глава 5. Критические ошибки приложений и операционной системы 639

5.1. Приложения, недопустимые операции и все-все-все.....	640
Доктор Ватсон.....	642
Отладчик Microsoft Visual Studio Debug.....	649
5.2. Обитатели сумеречной зоны, или из морга в реанимацию.....	650
Принудительный выход из функции.....	651
Раскрутка стека.....	654
Передача управления на функцию обработки сообщений.....	658
5.3. Как подключить дампы памяти.....	666
Восстановление системы после критического сбоя.....	676
Подключение дампа памяти.....	677

Глава 6. Формат PE-файлов 683

6.1. Особенности структуры PE-файлов в конкретных реализациях.....	684
6.2. Общие концепции и требования, предъявляемые к PE-файлам.....	686
6.3. Структура PE-файла.....	689
6.4. Что можно и что нельзя делать с PE-файлом.....	692
6.5. Описание основных полей PE-файла.....	695
[old-exe] e_magic.....	695
[old-exe] e_cpahdr.....	695
[old-exe] e_lfanew.....	696
[image_file_header] Machine.....	696
[image_file_header] NumberOfSections.....	696
[image_file_header] PointerToSymbolTable/NumberOfSymbols.....	697

<i>[image_file_header] SizeOfOptionalHeader</i>	697
<i>[image_file_header] Characteristics</i>	698
<i>[image_optional_header] Magic</i>	700
<i>[image_optional_header] SizeOfCode/SizeOfInitializedData/ SizeOfUninitializedData</i>	700
<i>[image_optional_header] BaseOfCode/BaseOfData</i>	701
<i>[image_optional_header] AddressOfEntryPoint</i>	701
<i>[image_optional_header] ImageBase</i>	702
<i>[image_optional_header] FileAlignment/SectionAlignment</i>	702
<i>[image_optional_header] SizeOfImage</i>	703
<i>[image_optional_header] SizeOfHeaders</i>	703
<i>[image_optional_header] CheckSum</i>	704
<i>[image_optional_header] Subsystem</i>	704
<i>[image_optional_header] DllCharacteristics</i>	705
<i>[image_optional_header] SizeOfStackReserve/SizeOfStackCommit, SizeOfHeapReserve/SizeOfHeapCommit</i>	706
<i>[image_optional_header] NumberOfRvaAndSizes</i>	706
<i>DATA_DIRECTORY</i>	707
Таблица секций.....	709
Экспорт.....	714
Импорт.....	718
Перемещаемые элементы	727

Глава 7. Техника внедрения и удаления кода из PE-файлов 731

7.1. Понятие X-кода и другие условные обозначения.....	732
7.2. Цели и задачи X-кода.....	733
7.3. Требования, предъявляемые к X-коду.....	736
7.4. Внедрение	737
Предотвращение повторного внедрения	738
Классификация механизмов внедрения	740
Категория А: внедрение в пустое место файла.....	741
Внедрение в регулярную последовательность байтов.....	750
Категория А: внедрение путем сжатия части файла	756
Категория А: создание нового NTFS-потока внутри файла	758
Категория В: растяжение заголовка	761
Категория В: сброс части секции в оверлей	763
Категория В: создание своего собственного оверлея.....	767
Категория С: расширение последней секции файла.....	768
Категория С: создание своей собственной секции	771
Категория С: расширение серединных секций файла.....	773
Категория Z: внедрение через автозагружаемые DLL	776

ПРИЛОЖЕНИЯ.....	777
Приложение 1. Разгон и торможение Windows NT.....	779
Структура ядра.....	780
Типы ядер	782
Почему непригодны тестовые пакеты	784
Обсуждение методик тестирования	786
Разность таймеров	787
Синхронизация	791
ACPI и IRQ	792
Переключение контекста.....	796
Длительность квантов	802
Обсуждение полученных результатов	805
Приложение 2. Практические советы по восстановлению системы в боевых условиях.....	807
Аппаратная часть.....	808
Оперативная память	809
Блок питания	811
И все-все-все.....	812
Приложение 3. Описание компакт-диска	815
Предметный указатель	817

Предисловие

Взлом — это естественная потребность всякого разумного существа. Тернистый путь познания истинной сути вещей проходит через их разрушение. Оглянитесь вокруг: физики-атомщики расщепляют ядра так, что брызги материи летят, химики-аналитики разбивают длинные молекулы на множество мелких, математики активно используют метод декомпозиции. И никто из них не заслуживает порицания!

Хакерство — это не вандализм. Это проявление природного любопытства к познанию окружающего нас мира. Дизассемблерные листинги, машинные команды, черные экраны SoftIce, напоминающие о первой молодости MS-DOS — все это безумно интересно и увлекательно. Посреди них раскинулся целый мир скрывающихся механизмов и защитных кодов. Не ищите его на картах — он существует лишь в обрывках беспорядочно разбросанных по полу распечаток, технических руководствах, автоматически открывающихся на самых интересных местах, и конечно же, многочисленных бессонных ночах, проведенных у монитора.

Это не учебник по взлому и не руководство по защите от хакеров. Таких книг уже написано предостаточно. Баста! Надоело! Перед вами лежат путевые заметки кодокопателя, своеобразный сборник любопытных историй, произведших с мышц'ем в киберпространстве. Вы побываете и внутри компиляторов фирмы Intel, и внутри защитных механизмов коммерческих программ, узнаете, как работает отладчик и как правильно держать его в руках. В общем, если не струсите и не отбросите эту книгу прочь, вас ожидает много интересного.

Об авторе

Небрежно одетый мышцх 28 лет, не обращающий внимания ни на мир, ни на тело, в котором живет, и обитающий исключительно в дебрях машинных кодов и зарослях технических спецификаций. Не общителен, ведет замкнутый

образ жизни хищного грызуна, практически никогда не покидающего свою норку — разве что на звезды посмотреть или на луну (повыть). Высшего образования нет, а теперь уже и не будет; личная жизнь не сложилась, да и вряд ли сложится, так что единственный способ убить время from dusk till dusker — это полностью отдаться работе.

Одержим компьютерами еще со старших классов средней школы (или еще раньше — уже, увы, не помню). Основная специализация — реинжиниринг (дизассемблирование), поиск уязвимостей (попросту говоря, "дыр") в существующих защитных механизмах и разработка собственных систем защит. Впрочем, компьютеры — не единственное и, вероятно, не самое главное увлечение в моей жизни. Помимо возни с железом и блужданий в непроходимых джунглях защитного кода, я не расстаюсь с миром звезд и моих телескопов, много читаю, да и пишу тоже (в последнее время как-то больше пишу, чем читаю). Хакерские мотивы моего творчества не случайны и объясняются по-детски естественным желанием заглянуть "под капот" компьютера и малость потыкать его ломом и молоточком, разумеется, фигурально — а как же иначе понять, как эта штука работает?

Если считать хакерами людей, одержимых познанием окружающего мира, то я — хакер.



О чем и для кого эта книга

If you do accept the society where we are compelled to live, its awfully egoistic way of life and its dirty "profit" values, you may eventually learn how to disable some simple protections, but you'll never be able to crack in the "right" way. You must learn to despise money, governments, televisions, trends, opinion-makers, public opinion, newspapers and all this preposterous, asinine shit if you want to grasp the noble art, coz in order to be emphatic with the code you must be free from all trivial and petty conventions, strange as it may sound. So you better take a good look around you... you'll find plenty of reasons to hate society and act against it, plenty of sparks to crackle programs in the right way... Hope all this did not sound too cretin.

+ORC an526164@anon.penet.fi

Изначально эта книга позиционировалась как хрестоматия для профессионалов, однако пробная публикация отдельных глав в сети профессионалам пришлось не по вкусу. Им не понравилось большое количество "воды" и слишком "разжеванные", с их точки зрения, объяснения. Начинающие кодокопатели резонно возражали, что для одного — "вода", для другого — хлеб, пиво и каша в придачу. Понятное дело, каждый читатель хотел видеть книгу такой, какая была бы наиболее удобна ему одному, но удовлетворить интересы всех категорий читателей в одной-единственной книге (к тому же, не претендующей на полноту и новизну излагаемой информации) — невозможно.

Основную ставку автор делает на начинающих — как на наиболее многочисленную и благодарную аудиторию. Профессионалы же вообще не нуждаются в подобных книгах. "Есть, — говорили они мне, — у тебя с десяток интересных страниц, но они размазаны по всему тексту, и потому читать такую книгу можно только по диагонали в порядке общего ознакомления". Нет, не подумайте, что такие заявления меня обидели! Напротив, помогли лучше понять свое место и предназначение.

Чего греха таить — до звания профессионала автору еще очень далеко, и потому позиционировать свои книги для той аудитории, к которой он не принадлежит, мягко говоря, нетактично. Правда, понятия профессионала и начинающего очень условны, и многие начинающие легко уделывают иных профессионалов. Племя подлинных профессионалов очень малочисленно и невелико — в прямом смысле слова, считанные единицы. Так что невоз-

можно сказать заранее: найдете ли вы что-то новое в данной книге или нет. Единственный способ выяснить это — купить ее и прочитать.

Эта книга не для кракеров! Несмотря на то, что в ней рассматриваются и даются в виде законченных технологий механизмы атак на широко распространенные системы, это просто информация, а не руководство к действию. В любом случае правовую ответственность за компьютерный вандализм еще никто не отменял, и прежде чем использовать полученные знания на практике, неплохо бы ознакомиться с уголовным кодексом и запастись адвокатом. В идеальном обществе принимаемые законы лишь закрепляют уже сложившуюся систему отношений, защищая при этом интересы большинства. А чего хочет большинство? Правильно! Хлеба и зрелищ! Ну, с хлебом все более или менее понятно. Даже в странах третьего мира от голода практически никто не умирает. Со зрелищами же ситуация значительно сложнее. На фоне катастрофической деградации аудио/видеоиндустрии борьба с пиратством приобретает воистину угрожающий размах, ущемляя интересы как отдельно взятых пользователей, так и всего мирового сообщества в целом. Власть захватили медиамагнаты, и демократия умерла. Сильные мира сего лоббируют законы, служащие паре десятков миллиардеров и противоречащие интересам остальных. Ряд научных и инженерных исследований в области информационной безопасности частично или полностью запрещен. Потребитель даже не имеет права заглянуть дизассемблером в тот продукт, который ему впаривают. Look, but don't touch! Touch, but don't taste! Taste, but don't swallow! Ситуация дошла до своего логического абсурда, и в воздухе запахло бунтом. Бунтом против тоталитаризма "демократического режима", бунтом против авторского и патентного права, когда один проницательный коммерсант отнимает у человечества то, что ему принадлежит по праву. Информация — общедоступный ресурс, такой же, как вода и воздух. Мы дети своей культуры. Наши мысли и суждения, которые мы искренне считаем своими, на самом деле представляют комбинацию уже давно придуманного и высказанного. Удачные находки, яркие идеи — все это результат осмысления и переосмысления когда-то услышанного или прочитанного. Вспомните свои разговоры в курилках — даже располагая диктофонной записью, невозможно установить, кто первым высказал идею, а кто ее подхватил. Знание — продукт коллективного разума. Никто не должен единолично им владеть.

Взлом защиты — это выражение протеста против насилия и несправедливости. Сажать за него можно, но бесполезно. Говорят, что в СССР одного человека посадили за то, что он сказал: "У нас нет демократии". А ведь взломщики и защитники информации не только враги, но еще и коллеги. Хакерство и программирование действительно очень тесно переплетены. Создание качественных и надежных защитных механизмов требует навыков низкоуровневой работы с операционной системой, драйверами и оборудованием; знаний архитектуры современных процессоров и учета особенно

стей кодогенерации конкретных компиляторов, помноженных на "биологию" используемых библиотек. На этом уровне программирования грань между собственно самим программированием и хакерством становится настолько зыбкой и неустойчивой, что я не рисковал бы ее провести.

Начнем с того, что всякая защита, равно как и любой другой компонент программного обеспечения, требует тщательного и всестороннего тестирования на предмет выяснения ее работоспособности. Под работоспособностью в данном контексте понимается способность защиты противостоять квалифицированным пользователям, вооруженным хакерским арсеналом (копировщиками защищенных дисков, эмуляторами виртуальных приводов, оконными шпионами и шпионами сообщений, файловыми мониторами и мониторами реестра). Качество защиты определяется отнюдь не ее стойкостью, но соотношением трудоемкости реализации защиты к трудоемкости ее взлома. В конечном счете, взломать можно любую защиту — это только вопрос времени, денег, квалификации взломщика и усилий, но грамотно реализованная защита не должна оставлять легких путей для своего взлома. Конкретный пример. Защита, привязывающаяся к сбойным секторам (которые действительно уникальны для каждого носителя), бесполезна, если не способна распознать их грубую эмуляцию некорректно заполненными полями EDC/ECC. Еще более конкретный пример. Привязка к геометрии спиральной дорожки лазерного диска, даже будучи реализованной без ошибок, обходится путем создания виртуального CD-ROM привода, имитирующего все особенности структуры оригинального диска. Для этого даже не нужно быть хакером — достаточно запустить Alcohol 120%, ломающий такие защиты автоматически.

Ошибки проектирования защитных механизмов очень дорого обходятся их разработчикам, но гарантированно застраховаться от подобных просчетов — невозможно. Попытка применения "научных" подходов к защите программного обеспечения — чистейшей воды фарс и бессмыслица. Хакеры смеются над академическими разработками в стиле "расчет траектории сферического коня в вакууме", и практически любая такая защита снимается за 15 минут без напряжения извилин. Вот грубый, но наглядный пример. Проектирование оборонной системы военной крепости без учета существования летательных средств позволяет захватить эту самую крепость чуть ли не на простом "кукурузнике" (MS WDB — кукурузник), не говоря уже об истребителях (SoftIce — истребитель, а IDA Pro — еще и бомбардировщик).

Для разработки защитных механизмов следует иметь хотя бы общее представление о методах работы и техническом арсенале противника, а еще лучше — владеть этим арсеналом не хуже противника (т. е. владеть им в совершенстве). Наличие боевого опыта (реально взломанных программ) очень и очень желательно — пребывание в шкуре взломщика позволяет досконально изучить тактику и стратегию наступательной стороны, давая тем самым возможность оптимальным образом сбалансировать оборону. Попросту

говоря, определить и усилить направления наиболее вероятного вторжения хакеров, сосредоточив здесь максимум своих интеллектуальных сил. А это значит, что разработчик защиты должен глубоко проникнуться психологией хакеров, настолько глубоко, чтобы начать мыслить, как хакер.

Таким образом, владение технологией защиты информации предполагает владение технологией взлома. Не зная того, как ломаются защиты, не зная их слабых сторон, не зная арсенала хакеров, невозможно создать стойкую, дешевую, и главное — простую в реализации защиту. Поэтому, описывая технику защиты, было бы, по меньшей мере, нечестно замалчивать технику ее взлома.

Существует мнение, что открытые публикации о дырах в системах безопасности приносят больше вреда, чем пользы, и их следует в обязательном порядке запретить. Другими словами, достойную защиту от копирования мы создать не можем, признавать свои ошибки — не хотим, и чтобы цивилизацию не разрушил первый же залетный дятел, мы должны перестрелять всех дятлов до единого. Совсем не собираясь апеллировать к заезженной поговорке "кто предупрежден — тот вооружен", обратимся за советом к фармацевтической индустрии. Как бы она отнеслась к появлению рекламы, пропагандирующей некий никем не проверенный, но зато невероятно эффективный препарат, исцеляющий немыслимое количество заболеваний и при этом обязательный к применению? Проводить химический анализ препарата вы не имеете права, равно как не имеете права открыто публиковать результаты своих исследований, выявивших, что за личиной "панацеи" скрывается обыкновенный и довольно низкокачественный аспирин с кучей посторонних примесей и целым "букетом" побочных эффектов. Еще бы! Ведь публикации подобного рода заметно охлаждают потребительский пыл, и предпочтение отдается другим препаратам.

Кто виноват: фирма, обманывающая своих покупателей, или исследователи, открывшие покупателям глаза? Если аналогия между фармацевтикой и программным обеспечением кому-то покажется некорректной, пусть он ответит на вопрос: какие задачи решают защитные механизмы и каким требованиям они должны отвечать? Всякая технология имеет свои ограничения и свои побочные эффекты. Рекламные лозунги, позиционирующие защиту как стойкую и абсолютно непрошибаемую, всегда неверны. Коль скоро носитель можно воспроизвести, можно его и скопировать. Весь вопрос в том — как? Запрет на хакерскую деятельность ничего не меняет. Тех, кто занимается несанкционированным клонированием дисков в промышленных масштабах, подобные запреты вряд ли остановят, а вот легальные исследователи будут страдать: профессиональный зуд, видите ли, дает о себе знать. Ну что поделаешь, есть на земле такая категория людей, что не может удержаться от соблазна заглянуть под крышку черного ящика и, дотрагиваясь до вращающихся шестеренок, пытается разобраться: как же все это, блин, работает? Специфика защитных механизмов состоит в том, что дерьмо визуально

ничем не отличается от шедевра. Вам остается уповать лишь на авторитет поставщика или же методично приобретать все продукты один за другим, при этом не будучи уверенным в том, что на рынке вообще присутствуют достойные защитные механизмы. И это действительно так! Качество коммерческих защит настолько низко, что они оказываются не в состоянии справиться с автоматическими копировщиками программ, запущенными обыкновенными пользователями, коих миллионы. Стоит ли говорить, что некопируемость автоматически копировщиками — это минимально разумное требование, предъявляемое ко всякой защите? В идеале же защита должна противостоять квалифицированным хакерам, вооруженным всем необходимым арсеналом программно-аппаратных средств взлома? Качественные защитные алгоритмы есть, но они не представлены на рынке. Почему? Да все потому, что отсутствие достоверной информации о стойкости тех или иных защитных пакетов не позволяет потребителю осуществлять сознательный выбор. А раз так — зачем производителям напрягаться?

Адептам авторского права важно понять: чем интенсивнее ломаются защитные механизмы, тем стремительнее они совершенствуются! Поскольку у разработчиков появляется реальный стимул к созданию действительно качественных и конкурентоспособных защитных пакетов! Начнем с малого: правило Кирхгофа — базовое правило для всех криптографических систем — гласит: стойкость шифра определяется только секретностью ключа. Криптоаналитику известны все детали процесса шифрования и дешифрования, кроме секретного ключа. Отличительный признак качественной защиты — подробное описание ее алгоритма. В самом деле, глупо скрывать то, что каждый хакер может добыть с помощью отладчика, ящика пива и дизассемблера. Собственно говоря, скрупулезное изучение подробностей функционирования защитного механизма можно только приветствовать. В конце концов существует такое понятие, как полнота представления сведений о товаре, и попытка сокрытия явных конструктивных дефектов, строго говоря, вообще не законна. Всякой Хорошей Вещи гласность только на руку, а вот Плохие Вещи боятся ее, как огня.

Апелляция к "цифровой эпохе" и якобы вытекающая отсюда необходимость пересмотра законов — это грязное словоблудие и ничего более. Большое количество судебных исков, вчиненных вполне легальным исследователям защищенных программ, не может не удивлять. Несмотря на то, что подавляющее большинство подобных исков решается мирным путем, сама тенденция выглядит довольно угрожающей. Чего доброго завтра и <Shift>, как "хакерскую" клавишу запретят, поскольку она позволяет отключить автозапуск лазерного диска в OS Windows, а некоторые защитные механизмы как раз на этом и основаны. То, что еще позавчера казалось фантазмагорическим бредом, вчера вылилось в реальный судебный иск.

Закон DMCA (Digital Millennium Copyright Art) действительно запрещает распространять технологии, устройства, продукты и услуги, созданные с целью

обходить существующие системы защиты, что выглядит вполне логично. Правозащитники пытаются уберечь мир от очевидных преступников и вандалов, однако, следует разделять взлом как таковой и исследовательскую деятельность в области информационной безопасности. Взлом, преследующий личную выгоду или совершаемый из хулиганских побуждений, достоин, по меньшей мере, порицания, а по большей — штрафа или тюремного заключения. Причем тяжесть наказания должна быть сопоставима с величиной причинного ущерба. Приравнивать хакеров к террористам, право же, не стоит! Терроризм, равно как и хакерство, демонстрирует катастрофическую неспособность правительства обеспечить должный уровень безопасности, подчеркивая чрезмерную сосредоточенность власть имущих на своих собственных интересах и проблемах. Все! Никаких других точек соприкосновения у террористов и хакеров нет!

Книги, рассматривающие вопросы безопасности исключительно со стороны защиты, грешат тем же, что и конструкторы запоминающих устройств, работающих только на запись: ни то, ни другое не имеет никакого практического применения. Но эта книга не предназначена для разработчиков защит! Я не хотел делать упор ни на одну из категорий, потому что проблемы защиты информации скорее организационные, чем технические. Разработчики ПО не нуждаются в таких книгах и вообще редко прислушиваются к советам по реализации защитных алгоритмов, а хакеры от всего этого просто балдеют и морально разлагаются, теряя стимул к развитию.

Термин "хакер" имеет множественные трактовки, которые было бы бессмысленно перечислять здесь и, тем более, останавливаться на какой-то из них, игнорируя остальные. Но в лучшем значении хакер — это индивидуал, смотрящий в корень и стремящийся разобраться во всем до конца. Для таких людей и предназначена эта книга. Везде, где это только возможно, я буду стремиться к обобщению и постараюсь не акцентировать внимание на конкретных реализациях. Это не означает, что в книге не встретится законченных схем. Напротив, в них не будет недостатка. Но я не ставлю перед собой цель снабдить хакера готовым инструментарием.

Я не буду пытаться научить читателей "ловить рыбу", а рискну пойти немного дальше и привить некоторые навыки самообучения. Из любой самой нестандартной ситуации и при самом скудном инструментарии всегда можно найти выход. Типовые схемы часто оказываются бессильными и бесполезными. Любые навыки слишком привязаны к конкретному окружению. В наше время больших перемен уже поздно хвататься за традиционные схемы обучения. На обучение просто нет времени. Большинству программистов приходится осваивать новые технологии на ходу, и задолго до конца обучения они уже полностью устаревают.

Системы защиты, заметим, развиваются заметно более медленно, и ничего принципиально нового за последние год-два не было придумано или широко внедрено. Это обнадеживает, особенно на фоне деградации качества реал-

лизации широко распространенных систем безопасности. С другой стороны, толковых учебников и изданий по данной тематике тоже не выходило. Это привело к тому, что действительно грамотно спроектированных и качественно реализованных защит сегодня на массовом рынке практически не наблюдается. Большинство авторов, с трудом припоминая школьный курс алгебры, разрабатывают собственные алгоритмы, которые при близком рассмотрении криптостойкими не являются. Все математическое богатство является бесхозным и нетронутым, хотя на рынке достаточно много качественных и вполне современных изданий. Я не стремлюсь повторить уже проделанную работу, а просто буду отсылать по ходу дела к конкретным источникам.

Цель этой книги — научить читателя самостоятельно добывать необходимые ему знания и навыки, порой не имея соответствующей литературы и информации. Современное информационное изобилие приводит к атрофированию навыков самостоятельного получения необходимых знаний. Парадоксально на первый взгляд, но недостаток литературы развивает и тренирует мозги куда лучше, чем ее избыток. Сам я осваивал ассемблер 8086 с помощью утилиты `debug.com`: кроме нее (и свободного времени) у меня в ту пору не было ничего. Логика работы команд изучалась анализом воздействия последних на регистры и память. Эта было утомительное занятие, и свободное владение ассемблером (без учета ряда некритичных команд) ко мне пришло приблизительно через три месяца. Наличие инструкции сократило бы этот срок до двух-трех дней (с учетом знания ассемблера других платформ), но зато не дало бы никаких полезных навыков.

Инструкция исключительно редко бывает исчерпывающей и доступной. Полученные давным-давно навыки актуальны для меня до сих пор, ибо тенденция обратной зависимости качества инструкции от ее объема в последнее время стала угрожающе превращаться из метафоры в реальность. Главный психологический барьер для многих — это ощущение беспомощности перед компьютером. Человек ощущает себя не хозяином ситуации, а незадачливым экзаменуемым, наобум пытающимся угадать, чего же от него хочет машина. Знакомая ситуация: вызываемая функция упрямо ведет себя не так, как описано в документации, и не работает.

В данном случае не помешает прибегнуть к дизассемблированию и анализу ситуации, но, возможно, оптимальным окажется другое решение. В любом случае возможны свои нетривиальные варианты. Наивно полагать, что схема "нажал на кнопку — получил банан" работает во всех ситуациях. Или что банан непременно удастся достать с помощью палки либо ящика. А если нет? Если задача не имеет известного и опробованного решения? Вот тогда и приходится действовать, как говорят, "по обстоятельствам". Научить этому читателя — вот моя задача.

Введение

История хакерства

I remember the good old days, when computers were mainframes, analysts were magicians, and programmers poned cards.

*Philip Fites, Peter Johnston, Martin Kratz.
"Computer viruses"*

Термин "хакер" прочно вошел в разговорный лексикон даже не имеющих никакого отношения к компьютеру людей. А его изначальное значение со временем забылось. Сегодня "хакер" синоним словам "бандит" и "компьютерный вандал". С другой стороны, предпринимаются неоднократные попытки "очистки" термина вводом новых понятий, например, "кракер" — коммерческий взломщик, в противовес якобы бескорыстным в своих побуждениях хакерам. На самом же деле, первые хакеры появились задолго до возникновения компьютеров, более того, задолго до зарождения цивилизации и даже появления человечества. Первым открытием хакеров было то удивительное свойство палки, которое давало возможность охоты, обороны и еще много чего одновременно. Нетривиальное решение задач, за гранью обычного восприятия мира — это главная черта хакеров. Им мало видеть предмет в трех измерениях. Для хакеров каждый предмет — это, прежде всего, объект со своими свойствами, методами и причинно-следственными связями.

На протяжении всей истории человечества всегда находились люди, которые выходили за рамки господствующих установок, традиций и создавали свою философию и субкультуру. По иронии судьбы хакерство оказалось тесно сплетенным с криминалом. Так было во все века, и так и будет до самого последнего вздоха человечества. Почему? Хакерская натура стремится разобраться во всем до конца, разрешить все до мельчайших подробностей, выйти за область определения объекта и проанализировать и испытать его поведение во всех нестандартных ситуациях. От простого смертного хакера, прежде всего, отличает исчерпывающее знание предмета. Абсолютное знание по умолчанию подразумевает абсолютную власть над системой. Очень трудно

устоять перед искушением и открывающимися перспективами. Между взломом компьютерной системы и механического сейфа нет принципиальной разницы, чтобы об этом ни говорили. Так что хакеры были всегда, и, по меньшей мере, спекулятивно — связывать их существование с ЭВМ и компьютерными технологиями.

Хакерское сообщество не монолитно в своих побуждениях, целях и мотивах. От невинной шалости до умышленного уничтожения информации очень большой путь. Невозможно осуждать человека за сам факт принадлежности к хакерам. Быть может, в его поступках и нет ничего дурного? К хакерам относят и компьютерных специалистов, занимающихся вопросами безопасности, поскольку им по роду своей деятельности приходится атаковать системы с целью обнаружить (и по возможности устранить) бреши в механизме защиты.

Существует громадное количество людей, интересующихся и влияющих на проблемы безопасности сетевых (да и не только) сообщений. Некоторые из них называют себя хакерами, но имеют скудный багаж знаний, нередко почерпнутый из популярных кинофильмов; другие же, досконально изучив все тонкости защит, могут и не считать себя хакерами, даже если совершают (или не совершают) регулярные атаки. И кто же в этой ситуации хакер, а кто нет?

Попытки ввода ценза на принадлежность к хакерам обречены на провал. Зачем усложнять суть? Хакер — тот, кто атакует систему. Как и зачем он это делает — предмет другого разговора. Человеческий язык направлен на упрощение и обобщение терминов. Любые искусственные построения сметаются временем. Плохо ли это, хорошо ли это — неважно. Никто не в силах изменить ход вещей окружающего мира, поэтому приходится жить по его законам.

Бытует мнение о существовании некоторых признаков принадлежности к хакерам. Это длинные (нечесанные) волосы, пиво, сигареты, пицца в неограниченных количествах и блуждающий в пространстве взгляд. Беглое исследование как будто бы подтверждает справедливость таких утверждений, однако подобные признаки являются не причиной, а следствием. Привязанность к компьютеру заставляет экономнее относиться к свободному времени, порой питаюсь всухомятку урывками и на ходу; крепкие напитки вызывают опьянение, затрудняющее мыслительную деятельность, поэтому компьютерные фанатики полностью или частично отказываются от них, переходя на пиво (а то и лимонад). Длинные волосы? Да, они свойственны всем компьютерщикам (и не только им), а вовсе не исключительно хакерам, как, кстати, и все другие "хакерские" признаки.

Еще до недавнего времени считалось: хакер по определению высококлассный специалист. Сегодня же можно атаковать систему, даже не имея никаких представлений о том, как она устроена. Достаточно воспользоваться

доступной информацией или готовой программной реализацией атаки. Поэтому к хакерам приходится причислять всех "кто хочет", потому что "не мочь" уже невозможно. Многие атакующие программы неплохо документированы и имеют простой, интуитивно понятный интерфейс, в большинстве случаев не требующий от "взломщика" ничего, кроме владения мышью.

Так ли велико различие между желанием и умением вторгаться в чужие системы? Кажется, желание в отсутствие умения бесполезно, но это не так. Знания возникают не сами по себе, а приобретаются в процессе обучения. И те, кто хочет, но не может, и те, кто может, но не хочет, — потенциальные злоумышленники, способные на противоположные действия.

Очередной исторический каприз: именно длительная компьютерная анархия позволила легально развиваться целому пласту субкультуры личностей, которые в других отраслях прочно ассоциировались с криминалом. До этого были радиолюбители, которые обходили электронные системы сигнализации, перехватывали секретные передачи и конструировали удивительные по своей природе устройства.

Между перечисленными категориями нет качественной разницы. Более того, обычно работа с компьютером связана с интересом к электронике, и обычное хакерское любопытство затрагивает не только компьютерные, но и любые другие системы защиты.

Я понимаю, что данная трактовка может встретить возражение и является ничем иным, как моим субъективным мнением. И тут мы приходим к любопытной лингвистической проблеме терминов. В самом же деле, любой термин можно описать, перечислив все свойства и лексические вхождения. Независимо от возможных трактовок термина объективная нагрузка на него может быть задана простым перечислением.

Развитие любого языка приводит к тому, что каждый термин достоверен только в момент его определения, после чего начинается размытие и расширение смысловой нагрузки, подминающей под себя изначальную идею создания. Обратимся к лучшему на сегодняшний день исследованию хакерской культуры, "Словарю Жаргона" Э. С. Рэймонда. Словарь гласит (перевод за читателем):

:hacker: [originally, someone who makes furniture with an axe] n.

1. A person who enjoys exploring the details of programmable systems and how to stretch their capabilities, as opposed to most users, who prefer to learn only the minimum necessary.
2. One who programs enthusiastically (even obsessively) or who enjoys programming rather than just theorizing about programming.
3. A person capable of appreciating {hack value}.
4. A person who is good at programming quickly.

5. An expert at a particular program, or one who frequently does work using it or on it; as in 'a UNIX hacker'. (Definitions 1 through 5 are correlated, and people who fit them congregate.)
6. An expert or enthusiast of any kind. One might be an astronomy hacker, for example.
7. One who enjoys the intellectual challenge of creatively overcoming or circumventing limitations.
8. [deprecated] A malicious meddler who tries to discover sensitive information by poking around. Hence 'password hacker', 'network hacker'. The correct term is {cracker}.

В рамках данной книги мы будем говорить исключительно о компьютерных хакерах. По непроверенным историческим данным, компьютеры, изначально созданные для узкоспециализированных военных задач, именно хакерами были восприняты, как платформы с безграничными возможностями. В то далекое время математики занимались исключительно гипотетическими машинами, которые имели очень отдаленное отношение к действительности. Воплощенные в груду металла инженерные идеи обогнали даже знаменитых математиков на десятилетия вперед. Предложенная дискретная архитектура была совершенна и привела к развитию соответствующей дискретной математики, большей частью описывая то, что инженеры давно воплотили в жизнь. Так изначально практические компьютерные технологии обогнали математические модели.

Сегодня нам трудно представить, что было время, когда компьютеры обслуживали сливки технической элиты, решая рутинные, поставленные задачи. Впрочем, для начальства не секрет, что свободное и не такое уж свободное машинное время использовалось для личных нужд и исследований персонала. Так и зарождалась субкультура людей, которые открыли в грохочущем монстре вторую Вселенную, свое второе Я. Чудовищное отставание нашей страны в области вычислительной техники, жесткая дисциплина, постоянные репрессии привели к тому, что субкультура хакеров, да и программистов, зародилась целиком в стенах лабораторий США и уже оттуда распространилась на весь мир. В результате даже наши отечественные субкультуры большей частью американизированы, особенно у нас, на фоне массового использования американской программно-аппаратной базы.

История происхождения термина "хакер"

Термин "кракер" (cracker, ломатель), был введен в 1985 г. самими хакерами в знак протеста журналистам, неправильно употребляющим, по их мнению, термин "хакер".

История же возникновения термина "хакер" доподлинно неизвестна. Ее истоки ведут к древнеанглийскому, в котором "хак" обозначал звук топора,

возникающий при ударе о дерево. Но звание хакера присуждалось далеко не всем, а лишь высококлассным столярам и плотникам, создающим едва ли не произведения искусства (другими словами, хакеры представляли собой одержимых плотников).

В шестидесятых годах прошлого столетия, а может быть и раньше, этот термин перекочевал в компьютерную среду. По одной из гипотез, звук "хак" приписывался перфоратору, прокалывающему бумагу; другие уверяют, что так клацали реле. Так или иначе, хакерами стали называть системных программистов, фанатично преданных своему делу, по аналогии с плотниками.

Традиция сохранялась на протяжении более десятка лет, затем новое поколение студентов, впервые увидевших компьютер, окрестило хакерами всех фанатиков без исключения, даже если те были заурядными пользователями.

Примечание

В Массачусетском технологическом институте и вовсе имеется свое, оригинальное понимание хакеров: "At MIT a hacker is someone who does some sort of interesting and creative work at a high intensity level. This applies to anything from writing computer programs to pulling a clever prank that amuses and delights everyone on campus".

Литературные произведения "The Shockware Rider" Джона Бруннера (John Brunner) 1975 г., "The Adolescence of P-1" Томаса Риана (Thomas Ryan) 1977 г. и наконец, знаменитый "Necromancer" Вильяма Гибсона (William Gibson), опубликованный в 1984 г., своими героями выбрали компьютерных взломщиков, бросающих вызов обществу. Это совпало с движением панков на западе и было молниеносно подхвачено молодежью. Сейчас ведутся жаркие споры: то ли представителей нового движения хакерами назвали журналисты, то ли те самостоятельно присвоили себе это звание, но с этого момента под хакерами стали подразумевать злоумышленников, мошенников и вандалов всех мастей, отчего легальные хакеры старого поколения по возможности пытались избегать величать себя этим титулом.

В настоящее время термин "хакер" стал поистине всеобъемлющ, отчего потерял всякую ценность и смысл. К хакерам относят откровенных бандитов, совершающих преступления с применением технических средств, просто мелких мошенников, вирусописателей, системных программистов, а порой и просто программистов, в особенности знающих ассемблер, экспертов по безопасности... Считают себя хакерами и те, у кого хватило таланта и способностей запустить готовую атакующую программу даже без осмысления принципа ее работы.

Психология хакера

Где бы ни организовывались вычислительные центры — в бесчисленных местах в Соединенных Штатах, как фактически во всех промышленных районах мира, можно наблюдать блестящих молодых людей, всклокоченных, часто с запавшими, но сияющими глазами, которые сидят за пультами управления вычислительных машин, сжав в напряжении руки в ожидании возможности пустить в ход свои пальцы, уже занесенные над кнопками и клавишами, приковывающими их внимание так же, как брошенная игральная кость приковывает взгляд игрока. Если они не находятся в таком трансе, то часто сидят за столами, заваленными машинными распечатками, которые они сосредоточенно изучают подобно людям, одержимым постижением кабалистического текста. Они работают, чуть ли не до полного изнеможения, по 20–30 часов подряд. Еду, если только они о ней заботятся, им приносят (кофе, кока-кола, бутерброды). Если возможно, они спят около вычислительной машины на раскладушках, но всего несколько часов, а затем — снова за пульт управления или к распечаткам. Их измятая одежда, немые и небритые физиономии, нечесаные волосы — все свидетельствует о том, что они не обращают внимания ни на свое тело, ни на мир, в котором живут. Они существуют, по крайней мере, когда они так увлечены, лишь в связи с вычислительными машинами и ради них. Они — "машинные наркоманы", одержимые программисты. Это явление наблюдается во всем мире.

Дж. Вейценбаум. "Возможности вычислительных машин и человеческий разум (от суждений к вычислениям)"

Типичный образ хакера конца девяностых прошлого века: молодой человек, так и не получивший систематического образования, открывает в компьютере свою собственную вселенную и уходит в нее целиком. "Как и в других областях человеческой деятельности, спектр отношения людей к програм-

мированию и вычислительным машинам очень широк: от ненависти, через полное безразличие до патологической привязанности или зависимости, которую можно квалифицировать как манию" — писал Николай Безруков в своей монографии "Компьютерная вирусология".

Но в компьютере, в отличие от других видов деятельности, все-таки есть нечто особенное. Легко ли вообразить себе человека, помешанного утром, днем и ночью пылесосить? Зато любителя проводить все свободное и несвободное время за компьютером представить несложно. Какое же это все-таки увлекательное занятие — писать программы! Какое наслаждение смотреть, как они работают, и как приятно видеть результаты прогонов! Это все и работой назвать язык не поворачивается — сплошные удовольствия.

Для объяснения этого феномена выдвинуто и выдвигается множество различных гипотез и предположений. В ход идут аргументы типа: компьютер это собеседник, общаться с которым невероятно интересно; виртуальные игровые миры дают те чувства свободы, силы, самоутверждения, власти, которых нам так не хватает в реальной жизни.

Среди отечественных психологов бытует мнение, что к компьютеру сильнее всего привязываются личности, по тем или иным причинам отвергаемые обществом. Это может быть физическое несовершенство, уродство или инвалидность. Лишенные нормального человеческого общения, эти люди тянутся к компьютеру, как уникальному средству самовыражения и самоутверждения. Впрочем, обратное утверждение не всегда справедливо — компьютерный фанатик не обязательно должен оказываться инвалидом.

Возможно, ближе всех к истине подобрались зарубежные психологи. Среди их пациентов нередко встречались лица, абсолютно не приспособленные к реальной жизни, испытывающие огромные трудности в общении с окружающими людьми, отличающиеся неадекватной реакцией на все происходящее, одним словом, внешне создающие впечатление умственных дегенератов, но вместе с тем превосходно (даже виртуозно) программирующих на компьютере. В большинстве случаев феномен объяснялся тем, что практически все такие пациенты страдали аутизмом.

Аутизмом (от латинского слова *autos* — "сам", аутизм — погружение в себя) называют тяжелое психическое расстройство, при котором больной самоизолируется и существует как бы вне контакта с окружающим миром, теряя способность формировать эмоциональные привязанности и строить общение с людьми. О причинах аутизма на сегодняшнем уровне развития медицины остается только гадать, но предполагают, что это связано с недоразвитием определенных долей мозга в сочетании с гиперразвитием остальных областей. Другой возможной причиной называют аномальный химический состав мозга, но, так или иначе, аутизм относят к врожденным заболеваниям, хотя временами по этому поводу высказываются серьезные возражения.

Процент заболеваемости колеблется от 4 до 15 случаев на 10 000 детей, значительная часть их которых — мальчики. По статистике только в одних Соединенных Штатах зарегистрировано более 400 000 аутистов, но у 80% из них показатели IQ выше среднего и нередко на уровне гениев.

"С одной стороны, я могу находить ошибки в программах так быстро, что людям становится неловко. Но я совершенно асоциален. Я не могу угадывать намерения и настроение человека по выражению его лица или по его жестам, различать социальные оттенки и, наверное, не умею пользоваться этим языком. У меня не сложились взаимоотношения ни с кем из моих коллег, я определенно существо не коллективное" — рассказывает о себе Петер Леви, один из основателей компании "Accent Technologies", страдающий этим заболеванием.

Аутисты испытывают затруднения в общении даже с близкими людьми, у них отсутствует интерес к окружающему миру, явно выражены страхи, особенности поведения. С самого детства, ощутившие себя несколько отстраненными от мира, от людей, затрудняющиеся в налаживании контактов со сверстниками, порой битые за свою непохожесть, они находят в компьютере отдушину, средство сравняться с другими людьми или превзойти их.

"В этом мире можно дать волю всем своим идеям и фантазиям — от детских соплей до изощренного садизма. Мир, который полностью зависит от своего могущественного властелина, безраздельно распоряжающегося жизнью и смертью любой твари, которой позволено жить в его мире. Уйти от забот и переживаний грубого материального мира, стать творцом и властителем это мечта большинства, но мечта потаенная, скрытая. Осуществление ее доступно немногим, лишь тем, кто хочет и может" — писал психолог Ю. П. Прокопенко в одной из своих статей, завершая ее таким напутствием: "Хоть с мясом отрывайте свой зад от стула перед компьютером, идите на воздух, общайтесь с людьми. Как бы ни была интересна задача, она не уйдет, а вот жизнь проходит... Если вы чувствуете себя гораздо увереннее в виртуальном мире, чем среди людей, попробуйте посоветоваться с психологом — вдруг чего дельного скажет. Его советы не обязательны для исполнения, но профессиональный опыт может подать вам вашу же проблему с такой неожиданной стороны, что изменится вся система жизненных оценок. Рискните, пусть будет у вас побольше того самого жизненного опыта, которого так не хватает аутисту при любой выраженности этой черты характера".

Попытка связать психическую патологию и компьютерную одержимость не является чем-то новым. Эту мысль отстаивал еще Дж. Вейценбаум в своей монографии "Возможности вычислительных машин и человеческий разум (от суждений к вычислениям)". Вот что он пишет по этому поводу: "Разложение, порожаемое всемогуществом программиста вычислительной машины, проявляется в форме, поучительной для сферы, значительно более обширной, чем мир вычислительной техники. Чтобы оценить его, придется обратиться к примеру психического расстройств, хотя и очень давно известного, но, по-видимому, преобразовавшегося благодаря вычислительным

машинам в новую разновидность — манию программирования". Предположение, что хакерство больше чем образ жизни, склад мышления и простая привязанность к компьютеру, многое проясняет и позволяет пересмотреть свои взгляды на, казалось бы, очевидные факты. Хакерами движет вовсе не желание навредить. Даже когда наглым образом вредят, они лишь стараются обратить на себя внимание, компенсируя недостаток общения. Этот бессознательный порыв может ими самими истолковываться стремлением отомстить обидевшему их человечеству, но это не всегда оказывается так. Человеческий мозг — невероятно сложный механизм, состоящий из множества обособленных скоплений нейронов, с трудом понимающих "язык" друг друга. Сознание — лишь надводная часть айсберга; большая же часть работы протекает на бессознательном уровне. Поэтому мотивы многих поступков так и остаются загадкой. Человек лишь пытается объяснить их так или эдак, зачастую ошибаясь в своих выводах.

Хотя аутистам и присущи внезапные вспышки агрессии, среди одержимых программистов вандализм встречается редко. "Нас обвиняют в том, что мы, мол, чувствуем себя самыми великими, ни во что не ставя людей, которые не работают с компьютерами. Чушь собачья. Как ни стараюсь вот сейчас представить, не получается у меня почувствовать превосходство над, допустим, слесарем потому, что я более или менее умею заставлять компьютер делать то, чего хочу я. Зато он гайки крутит так, как мне в жизни не суметь" — заметил некто по прозвищу Jen.

"Средства массовой информации обычно обращают внимание исключительно на негативные стороны хакерства (взлом и кражу информации, разработку и распространение вирусов), однако такой взгляд является односторонним: для хакеров характерно гипертрофированное увлечение познавательной деятельностью, направленной на выяснение закономерностей работы информационных технологий, что необязательно ведет к каким-либо асоциальным действиям. Наоборот, существует мнение, что многие удачные и полезные идеи в области программного обеспечения были в свое время выдвинуты и реализованы именно хакерами"— писала О. В. Смылова в своей работе "Методы полевого психологического исследования в сообществе хакеров".

Но все же не всегда хакерство оказывается следствием тяжелой патологии. Часто дело заключается в обычном юношеском максимализме, когда кажется: весь мир твой, и ты его хозяин на правах сильного. Отсюда же идет глубокое убеждение, что информация должна быть свободной, а программное обеспечение — бесплатным. В отличие от неизлечимого аутизма, юношеский максимализм с возрастом проходит: появляется работа, отнимающая все свободное (а у фанатиков и несвободное) время, вырабатывается профессионализм, и надобность доказывать окружающим, что ты не осел, исчезает. А вместе с ней исчезает и сам хакер.

Но не пытайтесь отождествить себя ни с какими портретами. Каждый человек уникален и не подлежит усредняющей классификации.

Лаборатория искусственного интеллекта и PDP-1

Нет четкой грани между богами и людьми: одни переходят в других.

Ф. Херберт. "Мессия Дюны"

Персонал, обслуживающий правительственные компьютеры, был серьезным, и машинное время жестко протоколировалось, поэтому хакерство в то время было настолько экзотической редкостью, что кануло в историю, не оставив за собой следа. Нам остается только гадать, а были ли в самом деле хакеры на машинах масштаба ЭНИАК? Будем надеяться, что были. Достоверно известно лишь то, что уже тогда частенько увольняли сотрудников за отклонение от технологических режимов работы аппаратуры. Являлось ли это следствием халатности или попыткой получить от машины больше, чем предусмотрел ее создатель, теперь уже никогда не будет известно. В 1954 г. был разработан первый массовый компьютер UNIVAC. Его стоимость была вполне доступной для приобретения рядом корпораций для своих нужд. Обычно это были крупные технологические и исследовательские центры. Впервые доступ к ЭВМ смогли получить реальные гражданские лица. Помимо основной деятельности, компьютер загружали бесполезными, но очень любопытными задачами. Появились первые головоломки и компьютерные парадоксы, такие как программа, которая распечатывает свое содержимое, или стирающая сама себя. Не представляя никакой материальной выгоды, они имели только познавательную ценность и способствовали развитию информатики, как отдельной дисциплины.

В это время всем уже было ясно, что будущее принадлежит этим электронным вычислителям, и обеспечить его должны были профессионалы. Но специалисты не возникают сами по себе из ничего. Для этого требуется обучение и солидная материальная база. Осознав это, американцы первые начали поставлять ЭВМ в лучшие институты страны. Первые компьютеры на основе процессора PDP-1 были размещены в подвалах учебных заведений в начале шестидесятых годов прошлого века. Отсюда и берет свое начало история хакерства. Но никто даже в Америке не мог предугадать, к чему это приведет. Груда электроники завоевала сердца тысяч студентов, и они были готовы сидеть за машиной круглыми сутками, отказывая себе в еде, сне, отдыхе. Эти люди и были теми первыми хакерами, оставившими после себя едва заметный, но все же еще сохранившийся исторический след. Социально сложилось так, что хакер в первую очередь, прежде всего, ассоциируется со студентом. По определению хакер это человек, располагающей уймой свободного времени, которое он проводит наедине с машиной. А что делать? Копание в недрах железа и операционных систем требует не только определенного склада ума и характера, но еще и времени.

Коммерческий программист, несмотря ни на какие наклонности, имеет очень мало шансов стать хакером. Работая по заказу, ограниченному жесткими временными рамками, просто безумие зарываться в дебри чужого кода или ценой бессонных ночей сокращать уже до предела оптимизированную программу на один байт.

Это не значит, однако, что все хакеры были исключительно студентами. В те времена темпы развития вычислительной техники здорово отличались от сегодняшних, и времени на разработку ПО отводилось достаточно для того, что бы познавательные эксперименты для удовлетворения собственного любопытства не были недостижимой роскошью. Особо отметим, что в те стародавние времена оптимизация была не только показателем меры крутости программиста, но и производственной необходимостью. Ограниченные памятью, скоростью и возможностями периферии на той технике могли программировать лишь гении.

Отметим, что на данном этапе хакерство ассоциировалось с изящностью программирования и профессионализмом. Компьютеры в те годы вызывали чувство глубокого уважения, и никто не мог представить, что когда-то появится такое малоприятное явление, как компьютерный вандализм. На машинах работал весь цвет интеллигенции. Это были культурные парни, и если они могли что-нибудь портить, так это по ошибке. На этом фоне заметно выделялась лаборатория искусственного интеллекта MIT (Массачусетский технологический институт, Massachusetts Institute of Technology) в Бостоне, в котором работали сливки компьютерной общественности. Со временем люди, которые там работали, разъедутся по разным городам и займутся другой работой, но они разнесут свой жаргон и стиль жизни, заражая им всех остальных. Но об этом несколько позднее.

Кроме MIT, центром хакерства в Америке был университет Корнелл (Cornell University), штат Нью-Йорк. Лояльное отношение к студентам и предоставление "ночного машинного времени" способствовало развитию интереса к вычислительной технике и формированию хакерских традиций. Это поколение навсегда оставило след в истории "звездными войнами", открыв тем самым возможности компьютера как игровой платформы. Первые образцы ANSI-искусства были созданы именно студентами Корнелла. И хотя это спорно, и действительно сегодня уже трудно установить, кто первый придумал применить символьный терминал для попытки передачи графического образа, несомненно, этот человек был хакером. Способность заставить систему делать то, чего никак не ожидал ее разработчик, поистине заслуживает восхищения.

В СССР, как в тоталитарном государстве, хакерство сформировалось с большим опозданием и явным заимствованием западного образа и культуры. Доступ студентов к компьютерам был жестко ограничен, и на фоне слабых кадров среди преподавателей отдельные "выскочки" так и остались в одиночестве. Последнее вообще стало национальной чертой русского ан-

деграунда. Отсутствие клубов и разобщенность людей не способствовали развитию собственной культуры.

Центрами возникновения отечественных хакеров стали не нищие институты, а производственные предприятия и НИИ. Не секрет, что в то время в журналы загрузки машинного времени писали "липу", а рабочее время уходило для удовлетворения собственного любопытства. Но вернемся в лабораторию MIT. Трудно сказать, почему именно здесь сложились благоприятные условия, и хакеры росли здесь, как на дрожжах. Еще труднее сказать, что заставляло многих людей, отказывая себе во всем, стремиться проводить ночи напролет вокруг грохочущих телетайпов, клацающих реле и перемигивающихся огоньков. Психологи пока отделяются молчанием, что же позволяет одним открывать в этой, вызывающей немое уважение, груде металла свою Вселенную, а другие в ней ничего кроме собственно металла, кремния и пластика увидеть не в состоянии? Почему сегодня трудно представить хакера, поклоняющегося системе Windows (хотя это очень хорошая система), но совершенно отчетливо мы видим хакера, прокалывающего дырки на перфокарте? Просто система Windows не является миром, который позволяет выражать собственное Я. В ее создании принимали участие сотни и тысячи людей, в результате чего система потеряла какую-то общую концепцию и напоминает старый чулан, забитый хламом, под которым погребено немало действительно блестящих и красивых инженерных решений.

Заслуга разработчиков PDP в том, что они смогли создать не только идеологически целостную, но и настолько гибкую архитектуру, что выражение программистской мысли превратилось в сущее удовольствие. По непроверенным данным, первый PDP-1 был предоставлен именно MIT. Тогда неудивительно, что именно тут сформировалась особая группа людей, которые были настоящими волшебниками больших машин. Язык определяет стиль мышления, поэтому мышление программиста постепенно преобразуется под воздействием используемой архитектуры в образ мышления ее создателя. Это настолько очевидно, что никто из психологов в этом не сомневается. Немного утрируя, можно сказать, что образованию хакерства таким, каким мы его знали в 60-х годах XX века, мы целиком и полностью обязаны человеку-легенде Кену Оулсену (Ken Olsen), основателю компании DEC и ведущему разработчику аппаратного обеспечения. Несомненно, он был хакером, и его образ невольно копировали все пользователи его детища свыше двух десятков лет. Действительно, нужны чертовски нетривиальные мозги, чтобы создать компьютер, который стоил всего \$120 000 против миллионных майнфреймов от IBM. Конечно, майнфреймы заведомо превосходили PDP, но эта машина имела неповторимую притягательность архитектуры, следы которой используется и до сих пор. Даже в наше время суперпроцессоров, базирующихся на ядре Pentium PRO (в котором реализованы позаимствованные из БЭСМ-6 идеи), немало людей упоительно программирует на ассемблере PDP, используя эмулятор или даже сохранившееся аппаратное обеспечение.

Я не хочу бросать камни в сторону других фирм, но шедевр PDP из них никто не повторил, да похоже не здорово к этому и стремился. Популярность PDP обеспечивал и тот факт, что это была машина профессионала и для профессионала. Низкая скорость компенсировалась изящностью и гибкостью выражения программистской мысли. Более того, именно низкие ресурсы послужили мощным толчком к необходимости глубокого изучения команд процессора и проведению бессонных ночей в поиске нужной комбинации пары сотен байтов, едва уместящихся в скудном объеме оперативной памяти. На больших машинах это попросту было бесполезно, и ресурсов хватало даже для выполнения достаточно кривого и необдуманного кода. Именно PDP оказалась тем удачным полигоном, в котором нашли свой рай компьютерные гуру. Такой успех способствовал росту поставок DEC, а значит и расширению фирмы, а вместе с ней полноводного хакерского течения, возникшего вокруг лабораторий крупнейших университетов.

Сегодня нам уже не понять психологию людей, работающих на машинах того времени. Концепция ввода-вывода, а значит, язык общения с машиной изменились коренным образом. Клавиатуру и терминал нельзя никак сравнить с тем воистину магическим процессом "слепого" ввода машинных кодов с отладочного пульта, или когда мигающий огоньками исполин заглывал очередную стопку перфокарт и после солидного периода задумчивости выдавал узкую ленту, распечатывающую полученный результат в двоичном виде. Люди в белых халатах, стерильный воздух... все это безвозвратно ушло в прошлое и унесло с собой культуру компьютерных жрецов. Сохранившиеся ее элементы стали идолами, утратив физическую сущность и контекст событий. Сегодня PDP-1 может быть редким музейным экспонатом, познавательной игрушкой в форме эмулятора или памятником инженерного гения, но живой машиной она не станет уже никогда. Ее дух ушел вместе с хакерами того времени, оброс легендами и приобрел тот античный оттенок, который появляется на всем, что уже много лет лежит без движения. Но ростки, пущенные PDP-1, дали всходы, которые не завяли и по сей день.

Сеть

Ива сгибается под ветром, пока не разрастется и не встанет стеной на его пути.
В этом ее предназначение.

Ф. Херберт. "Дюна"

До конца 60-х годов прошлого века хакеров можно было сопоставить с античными мастерами. Хак ассоциировался с высшим профессионализмом и вытекающей из него культурой поведения. Тесная связь культурного и интеллектуального уровней давно отмечалась психологами. И хотя это правило

не обходится без исключений, общей картины они не меняют. А картина была следующая: в полной замкнутости и отсутствии какой-либо связи между компьютерными центрами страны, каждый программист должен был получить необходимые ему знания сам. Это был долгий и тернистый путь. Информатика едва выходила из околонуточной комы, эффективные алгоритмы и приемы еще не были канонизированы, широкой программисткой общественности они были не известны. Как бы ни был очевиден древовидный поиск или линейная сортировка, до всего этого нужно было додуматься, и увы — далеко не всегда один раз. Отсутствие таких привычных для сегодняшнего поколения коммуникаций приводило к тому, что многие алгоритмы переоткрывались десятки раз в разных местах, прежде чем информация о последних успевала дойти до адресатов "естественным" путем — через книги и университеты.

Обмен знаниями происходил только в узких рамках университетских или лабораторных общин. В условиях такой тесной связи друг с другом компьютерный вандализм возникнуть просто не мог. "Паршивые овцы" быстро вычислялись и с позором выдворялись прочь. Да и было-то их немного. Машинное время и программистский труд в прямом смысле слова ценились на вес золота, и любая мысль о "завешивании" системы казалась кощунством, достойным сжигания на костре. В лексиконе тогдашних хакеров еще не появилось оскорбительное слово "ламер". Не то чтобы среди них не было таковых, просто программисты в то время намного лояльнее относились к непрофессионалам. Да и как можно было расценить такие оскорбления в тесных коллективах?

Сегодня, на фоне развитых сетевых коммуникаций, когда собеседники едва ли имеют шанс встретиться лицом к лицу, ситуация кардинально изменилась. Электронное общение принесло вместе с неоспоримыми благами не меньшую кучу мусора, с которым вынуждены жить сегодняшние хакеры. В 1969 г. по инициативе Управления перспективных исследований Министерства обороны США (Defense Advanced Research Projects Agency) создается первая вычислительная сеть, получившая название Advanced Research Projects Agency NETwork — ARPANET. ARPANET сразу объединила несколько университетов, находящихся в разных концах США, и стремительно продолжала расширяться. По очередной иронии судьбы эта сеть не планировалась для передачи секретных сведений, а просто для обмена открытой информацией и электронной перепиской, поэтому никаких серьезных разграничивающих доступ элементов в ее архитектуре не присутствовало. Это только лишний раз подчеркивает, что даже тогда о вандалах не только не имели представления, но даже не могли представить их появление в обозримом будущем.

Именно эта промашка, унаследованная общеизвестной сетью Интернет, привела к состоянию сегодняшней анархии. Но не будем пока забегать так далеко вперед. Сеть не только физически соединила компьютеры, но и ду-

ховно сплотила работающих на них людей. Хакеры, привыкшие к малоподвижному образу жизни, порой и не знали, где географически находится их противник, в соседней лаборатории или в другом штате. Интенсивное взаимодействие сотен и даже тысяч очень неглупых людей дало невероятный толчок прогрессу. Для тех времен было характерно открытое обсуждение технологий и инженерных решений. Любые недочеты быстро исправлялись, и программа (технология) уже в исправленном виде отправлялась в сеть на следующий цикл доработки.

Времена, когда каждый изобретал свой велосипед, уже уходили в историю. Вместо бессонных ночей, проведенных в попытках решения задачи, теперь проводили время в поисках уже готовой информации в быстро набирающей силы сети. Именно эта легкость получения уже готовой информации без необходимости понимания сути и послужила толчком к появлению в сети людей, которые сами ничего не делали, только копировали ресурсы других.

Число пользователей ARPANET продолжало расти, и не всех из них можно было назвать хорошими парнями. Кроме того, ресурсы сети, помимо собственно программистов, начали использовать и далекие от компьютеров люди для служебной и деловой переписки. Прежняя монолитность компьютерного сообщества рухнула. Теперь далеко не каждый человек, сидевший за терминалом, был программистом. Все чаще и чаще он оказывался, выражаясь сегодняшней терминологией, простым юзером (например, банкиром или бизнесменом). Успех ARPANET и совершенствование компьютерных технологий вели к тому, что ЭВМ превращалась в предмет массового спроса и потребления.

Хакерский жаргон становился знаком принадлежности к особой группе — касте компьютерных гуру. В то время хакерство еще не стало модным, и никому и в голову не приходило копировать их профессиональный сленг. Компьютерные технологии шагнули далеко вперед, заменив телетайпы терминалами, а перфокарты — клавиатурой. Не в меньшей мере это отразилось и на возросших вычислительных мощностях.

Ярким примером служат первые прототипы будущих вирусов — программы-кролики (rabbits). Не причиняя никаких разрушительных воздействий, они были сконструированы так, что, многократно копируя себя, захватывали большую часть ресурсов системы, отнимая процессорное время у других процессов. Возможно, они явились следствием программной ошибки, которая приводила к закликиванию и наделяла программу репродуктивным свойствам. Первоначально кролики встречались только на локальных машинах, но с появлением глобальной сети быстро "научились" размножаться и там. В конце 60-х годов прошлого века была обнаружена саморазмножающаяся программа по сети ARPANET, известная как Creeper (Вьюнок). Вьюнок проявлял себя текстовым сообщением

I'M THE CREEPER ... CATCH ME IF YOU CAN,

и экономно относился к ресурсам пораженной машины, не причиняя никакого вреда, кроме легкого беспокойства. Каким бы безвредным Вьюнок ни казался, он показал, что проникновение на компьютер возможно без ведома и против желания его владельцев.

С появлением Сгеерг появились и первые системы защиты. Да, теперь компьютеры превратились в ту ценность, которую следовало охранять не только от воров с отмычками, но от злоумышленных разрушительных команд, проникающих по сети или на магнитных носителях.

Первым шагом в борьбе против Вьюнка стал Жнец (Reaper), репродуцирующийся наподобие Сгеерг, но уничтожающий все встретившиеся копии последнего. История скрывает от нас, чем закончилась борьба двух программ. От подобного подхода к защите в последствии отказались. Однако копии обеих программ еще долго бродили по сети.

Уже в 80-х останутся лишь отдельные одиночки, хакерство как общественное движение прекратит свое существование. Но мы слишком забежали вперед. Вернемся вновь к концу 60-х — началу 70-х годов прошлого века.

Си и UNIX

"Такие были времена. Я, например, к этому времени освоил около 15 ассемблеров и кучу ненужных машин" — писал Вадим Антонов в своих воспоминаниях. Пару десятков лет назад аппаратные ресурсы были катастрофически ограничены, и программисты работали большей частью в машинных кодах. Сначала текст программы составляли на бумаге (!), тщательно проверяли, переносили на перфоленту и "относишь колоду карточек этак на 500, кладешь на полку. Через день (а если повезет, то и через час) на полке появляется распечатка" — вспоминает Вадим Маслов.

Огромным достижением стало изобретение ассемблера, позволяющего абстрагироваться от неудобного для человека машинного кода. Все команды получили легко запоминающиеся символьные имена — мнемоники, а большую часть работы по вычислению адресов переходов и смещений компьютер взял на себя. Но за удобства пришлось заплатить — программа, написанная на ассемблере, требовала перевода в машинный код перед запуском — ассемблирования. Непосредственное общение с ЭВМ утрачивалось, а программисты изгонялись из машинных залов, уступая свое место оператору. Поэтому многие представители старого поколения крайне негативно относились к новинке прогресса, считая программирование на ассемблере "ненастоящим".

Кстати, неверно думать, что раньше и вовсе не существовало высокопроизводительных компьютеров. Так, например, компания Honeywell в 1973 г. приступила к выпуску многопроцессорных компьютеров, оснащенных в стандартной конфигурации 768 Кбайт ОЗУ и дисковым накопителем 1,6 Гбайт.

Стоило это удовольствие порядка семи млн долларов, но быстро окупалось дешевым программным обеспечением, которое уже не умирало при переходе на другую машину.

Но такие компьютеры были доступны лишь крупнейшим институтам и фирмам. Производители еще тогда предчувствовали закон Мура, официально сформулированный в 1978 г., и в лице компаний Bell Labs, General Electric's, Ford и MIT в 1965 г. вплотную занялись дорогостоящими экспериментами, целью которых было создание универсальной, переносимой, многопользовательской, высокопроизводительной операционной системы.

Для этого проекта General Electric пожертвовала высокопроизводительной 36-разрядной машиной GE-645 (с неплохим и по сегодняшним меркам процессором), оснащенной превосходной канальной подсистемой ввода/вывода.

Проект получил название MULTICS (Multiplexed Information & Computing Service). Немногим позже, в апреле 1969 Bell Labs разочаруется в достигнутых результатах и прекратит свое участие в проекте, сочтя его неудачным, но идеи, заложенные в MULTICS, найдут применение в операционных системах RSX, VMS, UNIX и даже Windows NT. В отличие от своих предшественниц, MULTICS разрабатывалась на интерпретируемом языке высокого уровня PL/1, созданном на основе языков ALGOL, FORTRAN и COBOL и ориентированном, в первую очередь, на задачи моделирования. Это довольно развитый язык, поддерживающий работу со списками и другими сложными структурами данных и допускающий выделение памяти под переменные различными способами. Но каким бы вычурным и многословным ни был синтаксис PL/1, писалось на нем намного быстрее, чем на ассемблере, и к 1968 г. (т. е. спустя три года после начала проекта) MULTICS начала обретать черты законченной операционной системы.

Сдерживаемые катастрофическим недостатком оперативной памяти, разработчики додумались до виртуальной памяти со страничной организацией, широко используемой сегодня в таких операционных системах, как UNIX и Windows. Виртуальная память имела сегментно-страничную организацию, отделяя сегменты данных от программного кода. Все сегменты имели атрибуты защиты, определяющие привилегии доступа. Перед каждой попыткой чтения/записи данных или исполнения кода чужого сегмента операционная система проверяла наличие прав на такую операцию, гарантируя надежную защиту критических участков кода от посягательств злоумышленников или некорректно работающих программ. К слову сказать, ни UNIX, ни Windows не обеспечивают подобной многоуровневой защиты. Отделяя прикладные приложения от ядра операционной системы, они в то же время позволяют уронить это самое ядро некорректно написанным драйвером, имеющим равные с ядром привилегии.

Именно в MULTICS впервые появилась возможность динамического связывания модулей в ходе выполнения программы, более известная современному читателю по пресловутым DLL в Windows. Такой прием логически завершил эволюцию совершенствования оверлеев, обеспечив единый, унифицированный интерфейс для всех программ, позволяя сэкономить значительную часть оперативной памяти и процессорных ресурсов. Один и тот же модуль (например, подпрограмма вывода сообщений на экран) теперь динамически загружался с диска и мог использоваться несколькими приложениями одновременно. Правда, при такой организации возникали проблемы совместного использования библиотек. Допустим, некое приложение, загрузившее для своих нужд динамическую библиотеку и считающее ее "в доску своей", в действительности оказалось отосланным к уже загруженному в память сегменту, активно используемому и другими приложениями. Что произойдет, если приложение, считающее библиотеку своей, попытается ее слегка модифицировать (при условии, что необходимые права у него есть)? Разумеется, незамедлительно грохнутся все остальные приложения, для которых такой поворот событий окажется полной неожиданностью. Поэтому разработчики придумали механизм "копирования при записи" — при первой же попытке модификации коллективно используемого сегмента создается его копия, предоставляемая в полное распоряжение модифицирующему коду.

Иерархическая файловая система впервые появилась именно в MULTICS, а не в UNIX, как пытаются утверждать некоторые. Файловая система MULTICS не только допускала вложенные директории, но и объединяла в одну логическую древовидную структуру файлы, физически расположенные на разных носителях. На уровне реализации это выглядело двоичным деревом, в узлах которого находились именованные каталоги, а листьями выступали ссылки на файлы. Современные операционные системы UNIX и Windows используют упрощенный вариант такой схемы.

А проецируемые в память файлы (memory mapped files) родились вовсе не в Windows NT, а в том же MULTICS. Традиционно файл читался в память, а если этой памяти оказывалось недостаточно, считанные фрагменты вновь сбрасывались на диск. Кому-то из разработчиков MULTICS это показалось слишком неэкономичным, и он предложил спроецировать файл в виртуальную память, а затем и вовсе объединить подсистему ввода/вывода с менеджером виртуальной памяти. Таким образом удалось просто и элегантно сократить число обращений к диску, попутно выкинув часть дублирующего кода из операционной системы.

Оконный интерфейс, обособленный в отдельную подсистему, также впервые появился в MULTICS. Конечно, ни о какой графике и мыши речь еще не шла, но взаимодействие с пользователями даже по современным понятиям было достаточно удобным и наглядным.

Но, помимо очевидных успехов, не меньше было и недостатков. Система оказалась необычайно прожорлива и для эффективной работы требовала

оборудования астрономической стоимости. Даже с учетом снижения цен на компьютеры рынок потенциальных покупателей был смехотворно мал. Практически единственным пользователем MULTICS оказалась компания Ford. Остальные были не в состоянии выложить требуемую сумму (к тому же, платить приходилось не только за "железо", но не в меньшей степени и за операционную систему).

Видя все это, руководство Bell Labs посчитало свое дальнейшее присутствие в проекте бессмысленным и в 1969 г. вышло из него. В MIT продолжали совершенствовать систему и к октябрю того же года довели ее до законченного состояния, но, как и предрекала Bell Labs, своего покупателя система не нашла и осталась невостребованной.

С этого момента и начался отсчет истории системы UNIX. Объявив о прекращении участия в проекте, Bell Labs отозвала всех своих разработчиков, среди которых оказались Деннис Ритчи, Кен Томпсон, Мак Илрой и Джон Осанна. Движимые желанием использовать накопленный опыт для создания дешевого и нетребовательного к аппаратным ресурсам усеченного варианта MULTICS, они обратились к администрации руководства Bell Labs с просьбой приобрести для этой цели компьютер среднего класса и выделить некоторую сумму под проект. Однако компания, разочарованная провалом MULTICS, отказалась финансировать эту затею. Сейчас все больше историков сходятся на том, что формулировка проекта выглядела недостаточно убедительной и неаргументированной. По другому мнению: Bell Labs просто охладела к операционным системам и не видела в них никакого источника прибыли — одни расходы.

Однако отказ ничуть не смутил разработчиков. И Томпсон вместе с Ритчи и Кэнадаем приступили к проектированию файловой системы будущей операционной системы на бумаге! В процессе этого занятия в голову Томпсона пришла блестящая мысль: объединить подключенные к компьютеру устройства вместе с файлами в одну иерархическую систему. Переполненный желанием испытать свою идею на практике, он обнаружил в одном из "пыльных углов фирмы" редко используемый PDP-7 и получил разрешение руководства позаимствовать его во временное использование. Наученный горьким опытом, Томпсон ни слова не упомянул об операционной системе и объяснил свою потребность в компьютере желанием перенести на него игровую программу Space Travel (Космическое Путешествие), написанную им в том же 1969 г. в ходе проекта MULTICS на языке FORTAN под операционной системой GECOS (стандартной ОС компьютеров General Electric). В то время к компьютерным играм относились куда серьезнее, чем сейчас, и заверения Томсона, что, переписав ее на ассемблер, он добьется значительного увеличения производительности, склонили руководство к временному выделению техники.

К сожалению, для PDP-7 не существовало ни приемлемого ассемблера, ни библиотек для поддержки вычислений с плавающей точкой (а они требова-

лись для игры). Поэтому Томпсон использовал кросс-ассемблер GECOS, умеющий формировать ленты, читаемые PDP-7, и создал необходимый инструментарий самостоятельно. В дальнейшем вся работа велась исключительно на компьютере PDP-7 без поддержки со стороны GECOS.

Как нетрудно догадаться, в первую очередь Томпсон приступил к экспериментам со своей новой файловой системой и с удивлением обнаружил: операции ввода/вывода значительно упрощаются, а программирование игры ускоряется. Параллельно с написанием игры создавался набор вспомогательных утилит для копирования, удаления, редактирования файлов и даже примитивный командный интерпретатор. В начале 1970 г. все это хозяйство было уже достаточно хорошо отлажено и даже ухитрялось сносно работать. Не было ни мультизадачности, ни продуманного и эффективного ввода/вывода, ни достойной организации процессов, но все это работало, а созданный программный инструментарий оказался удобным и достаточно мощным.

С легкой руки Брайна Кернигана, новая система в пародию на MULTICS получила название UNICS (Uniplexed Information & Computing Service). Позже программисты с нестандартным мышлением, склонные к сокращениям и оптимизации, заменили CS на X, и система приобрела название UNIX.

Но время, отведенное Томпсону, подошло к концу, и компьютер PDP-7 пришлось возвращать. Осанна предложил руководству вместо операционной системы финансировать систему подготовки текстов и патентов, в которой компания крайне нуждалась. Уловка удалась, и вскоре специально для разработчиков был приобретен новейший по тем временам компьютер PDP-11 стоимостью в 65 тыс. долларов, располагающий 24 Кбайт оперативной памяти и 512 Кбайт накопителей (впрочем, накопителей к нему еще не существовало). Перенос UNIX на новую платформу не представлял сложности (архитектуры обоих компьютеров были близки), но несколько затянулся по причине отсутствия накопителей для PDP-11. Когда же они появились, система была без проблем перенесена. Ко второй половине 1971 г. UNIX начала использоваться в патентном бюро, значительно превосходя в удобности и мощности аналогичные имеющиеся на рынке системы. Поэтому руководство дало добро на развитие проекта, и коллектив разработчиков сосредоточил все усилия на дальнейшем совершенствовании системы.

Перенос UNIX с PDP-7 на PDP-11 заставил разработчиков задуматься над путями повышения мобильности. К тому же, уж очень не хотелось вновь корпеть над ассемблером. Некоторые даже порывались писать новую систему на PL/1, но это значительно ухудшило бы производительность и вряд ли заслужило бы одобрение руководства. В качестве разумной компенсации предлагалось выбрать FORTAN или новый язык Би — один из диалектов BCPL. Би привлекал простотой и легкостью изучения, наглядностью листингов и неплохой производительностью. Так, в конце концов, выбор остановили на нем. Поскольку никакой реализации Би для платформы PDP-11

еще не существовало, Томпсону пришлось самостоятельно разрабатывать интерпретирующую систему.

Вторая версия UNIX появилась в 1972 г. Главным нововведением стала поддержка конвейера (pipe), позаимствованная Мак'Илроем из операционной системы DTSS (Dartmouth time-sharing System). Конвейеры обеспечивали простой и элегантный обмен данными между процессами даже в однозадачной среде и позволили сделать еще один революционный шаг вперед (кстати, конвейеры поддерживаются практически всеми современными операционными системами, в том числе и MS-DOS).

Использование интерпретируемого языка заметно ухудшило производительность системы, а в процессе работы выявились многочисленные недостатки, присущие Би. Самый неприятный из них — отсутствие типов переменных (точнее говоря, поддерживался всего один тип, равный машинному слову). Постоянные же преобразования с помощью специальных библиотечных функций порождали множество трудноуловимых ошибок. Когда всем это окончательно надоело, Деннис Ритчи, увлекающийся разработкой языков, решил усовершенствовать Би и добавил в него систему типов. Новый язык получил название Си, согласно второму символу в BCPL. Для улучшения производительности Томпсон предложил Ритчи написать компилятор, переводящий программы, написанные на Си, в машинный код.

Очередная версия UNIX отличалась завидной производительностью, практически не уступая версии, написанной на ассемблере, но потребовала значительно меньше усилий для своего создания и не была связана с какой-то одной конкретной архитектурой. Из 13 000 строк программы операционной системы лишь 800 принадлежали низкоуровневым модулям, написанным на ассемблере.

И хотя Си первоначально ориентировался на систему UNIX, он быстро завоевал популярность и на других платформах. Вскоре появились реализации для IBM SYSTEM/370, Honeywell 6000, INTERDATA 8/32. Си завоевал признание программистов, использующих его для решения самых различных задач, как-то: низкоуровневое и высокоуровневое системное программирование, встраиваемые системы, финансовые и научные расчеты, общее прикладное программирование. При всех присущих ему недостатках (сходите на **SU.C-CPP**, почитайте Юрия Харона) это лучшее средство для выражения программистской мысли в наиболее естественной для нее форме. Конструкции в стиле:

```
x == (flag?sin:cos)(y)
```

здесь вполне законны и являются нормой. В этом смысле Си очень похож на спектрумовский Basic, везде допускающий подстановку выражений, где это только возможно. 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 Си категорически не рекомендуется. Но ведь пишут же. Отсюда берутся черви, атаки на удаленные системы и прочие коварства виртуального мира. Си не является прикладным языком. Для работы с ним нужно тренированное системное мышление. Ничто не убережет программиста от неверных указателей и выхода за рамки массивов. Никто не позаботится завершить строку разделяющим нулем, пока программист это явно не укажет в программе.

Си находится в совсем другой плоскости, нежели ассемблер, и никак не освобождает истинных хакеров от знаний последнего. "Родной" язык не может быть вытеснен никакой другой высокоуровневой прослойкой. Пусть он потерял актуальность с появлением Си и быстродействующих процессоров, но в программировании непосредственно железа есть то магическое очарование, которое толкает многих на его изучение даже сегодня.

Теперь структурное программирование считается достоянием истории. В моду вошел ООП. Си++ завладел умами программистов. Объектно-ориентированный подход пропагандируется как единственно возможный способ программирования вообще, и на приверженцев классического Си смотрят, как на чудачков или недоучек. Прямо насилие какое-то получается! На самом деле преимущество ООП перед 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 ведь Си++ поддерживает не одну, а целых три парадигмы программирования: структурное программирование в духе улучшенного Си, абстрактные типы данных, позаимствованные из ADA и, наконец, объектно-ориентированный язык в стиле Simula. Вот что говорит по этому поводу Бьерн Страустрап: "При создании программы всегда есть некоторое количество вариантов, но в большинстве языков выбор сделал за вас проектировщик языка. В случае Си++ это не так — выбор за вами. Такая гибкость, естественно, непереносима для тех, кто считает, что существует лишь один правильный способ действий. Она может так же отпугнуть начинающих пользователей и преподавателей, полагающих, что язык хорош, если его можно выучить за неделю. Си++ 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 понадобится".

Приплюснутый Си — это целый мир. Богатый ассортимент языковых возможностей еще не обязывает ими пользоваться. Объектный подход бесполезен в драйверах. Сколько программисты ни пытались найти ему применения — так и не получилось, а вот парадигму "улучшенного Си" (объявление перемен-

ных по месту использования, а не в начале программы) используют многие. Правда, в драйверах (равно как и в модулях сопряжения со средой) жесткая типизация приплюнутого Си порождает дикий кастинг (casting, явное преобразование типов), уродуя исходный код и отнимая массу времени. Автоматической сборки мусора в Си++ 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 с завидной регулярностью встречаются и до сих пор. Так что Си++ это не панацея, а всего лишь раздутая рекламная кампания. Страуstrup оценивал количество пользователей приплюнутом Си в 5 тыс. программистов по всему миру. Наверяд 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.

Однако мы опять забегаем вперед. Создатели UNIX, разработавшие ее для своих нужд, вряд ли представляли себе, какого джина из бутылки они выпускают. Язык формирует мышление, и эти два хакера определили мышление миллионов людей, по крайней мере, на десятилетия вперед. Не секрет, что продукты "внутрифирменного использования" очень часто представляют собой жалкое зрелище, поскольку работодатели редко позволяют вкладывать деньги в изначально некоммерческий продукт. В свете этого шедевр UNIX кажется еще больше поразительным и необъяснимым. Система не только вобрала в себя лучшие по тем временам многопользовательские решения, но и была устойчива в работе и неплохо защищена. Разграничение затрагивало и системные ресурсы. Именно UNIX послужила тем барьером, который остановил растущие попытки закидывания кроликов в сеть, и на несколько лет компьютерный мир вновь обрел устойчивость.

Успех системы позволил AT&T значительно усилить свои позиции на рынке. UNIX, в противоположность системе Windows 90-х годов, не только был удачной платформой, но и дружелюбной к профессионалу средой, развивающей абстрактное мышление и способствующей развитию.

Конец хакеров шестидесятых

Я не должна бояться. Страх убивает разум. Страх — малая смерть, которая приносит полное уничтожение. Я смотрю в лицо моему страху...

Ф. Херберт. "Дюна"

Перешагнем на пару лет вперед. Эти годы прошли в непрерывном совершенствовании технологий программирования и аппаратного обеспечения.

Росли вычислительная мощь ЭВМ, емкость периферийных накопителей и быстродействие центрального процессора. Собственно, в то время все упиралось больше в деньги, нежели в технологии. Если клиент хотел решать с помощью компьютера серьезные задачи, то получение необходимой машинной мощности было только проблемой его кошелька. Производители компьютеров заметно обгоняли в те годы потребности клиентов.

И необходимость в высокопрофессиональных программистах и тщательно оптимизированном коде мало-помалу начала отпадать. Ярким подтверждением тому была UNIX, полностью написанная на компилируемом языке. Могли ли представить программисты такое расточительство хотя бы пару лет назад? Си стал первым кирпичом в фундаменте быстрых средств проектирования, который сегодня трансформировался в системы наподобие Visual Basic и DELPHI. Время обучения программистов резко сократилось, и профессионалы становились просто не нужны. Это вызвало приток специалистов в фирмы, специализирующиеся сугубо на программном обеспечении. А таковые появились во множестве, поскольку UNIX стала кроссплатформенной системой, и программы, написанные для одной модели компьютера, могли с минимальными переделками работать на любой другой.

И тут началось самое интересное. Каждый стремился обогнать другого. Это соревнование проходило не безболезненно, в первую очередь, страдал код, вернее, его качество. Руководство фирм вынуждало инженеров использовать все пути удешевления и ускорения разработки.

Если бы этим все и ограничилось, было бы полбеды. И код, написанный настоящим хакером даже в состоянии сумбурной спешки, всегда несет в себе долю оригинальности и собственного "я" программиста. Однако, стремясь максимально эффективно использовать труд людей, руководство разделяло коллектив на небольшие группы из трех-пяти человек, каждый из которых решал свою локальную задачу, зажатый тесными рамками остальных.

Творить и показывать свою индивидуальность в этих условиях становилось попросту невозможным. И профессионалы начали потихоньку разбегаться. Кто пытался организовать свою фирму, кто вообще уходил из программирования, будучи не в силах переломить ситуацию. Одним словом, сплоченному хакерскому сообществу шестидесятых наступил конец. Хакеры уходили, а вместе с ними, как туман, рассеивалась их культура. Уже никогда не повторится время больших машин, время глубокого эмоционального подъема и задиристого исследовательского духа, когда компьютеры еще не были коммерческой игрушкой.

Коммерция и жажда наживы поглощали на своем пути все. Главным критерием становилась скорость написания продукта. И не важно, какого он был качества. Любые недостатки и ошибки программного обеспечения не обнаруживаются сразу, и всегда есть время их устранить... Собственно, под сопровождением ПО почти любая фирма понимала его доработку и доведение до некоторого подобия законченного продукта.

Быть может, я сгущаю краски? Каждый имеет право на свою точку зрения. И нужно быть большим оптимистом, чтобы увидеть хоть что-то хорошее в конкурентной борьбе, спекуляции старыми идеями и технологиями, которая просто вышвырнула многих специалистов на обочину дороги. "Крутой поворот" (как любит выражаться Билл Гейтс) на этот раз и правда оказался настолько крутым и непредвиденным, что никто даже не успел к нему как следует подготовиться. Весь научно-технический потенциал в то время не охранялся семью печатями и мог свободно использоваться кем угодно и как угодно. Этим-то и воспользовались коммерсанты. Зачем было вести свои разработки, когда имелись уже готовые?

Профессионалы, инженеры и ученые предвидели, что существующие технологии исчерпают себя в ближайшие несколько лет. Разумеется, компьютерные магнаты также предвидели это и были не в восторге. С некоторым запозданием они все же начали спонсировать фундаментальные исследования и нанимать на работу талантливых инженеров, от которых требовалось только одно — передовые технологии.

К сожалению, инженерный персонал был поставлен в довольно жесткие рамки. Однако некоторая свобода творчества оставалась, которая, впрочем, досталась лишь "верхушке" разработчиков — архитекторами и идеологам проекта. С другой стороны, в этом был определенный здравый смысл. Рядовые инженеры являлись лишь инструментом в руках архитекторов, и чем послушней был этот инструмент, тем удобнее им было управлять.

Невозможно однозначно сказать, плохо это или хорошо. Новое время устанавливало свои новые законы, сменяя старые устои и ломая судьбы десяткам тысяч людей. С другой стороны, такая организация труда позволяла получать качественно новые результаты, и новые технологии стали плодиться, как грибы после дождя. Научный прогресс пополз вверх, опровергая бытующий тезис, что наука и творчество едины. Оказалось, что они даже не взаимосвязаны.

Хакеры шестидесятых вырождались. Не было больше больших машин и открытого духа свободного общения. С каждым днем все больше и больше находок и неординарных решений окутывались мраком коммерческой тайны и тщательно охранялись фирмой от окружающих. Красивые решения уже никому не были нужны и никого не интересовали. Мощность компьютеров того времени была уже достаточна, чтобы небрежно написанный код удовлетворял заказчика по всем параметрам (скорости, объему). Железо дешевило куда быстрее программного обеспечения. Легче было купить четверо более мощный компьютер для дешевой небрежно написанной программы, чем приобретать для дешевой машины оптимизированный до последнего байта код, стоимость которого в этом случае была бы просто астрономической. Так, к примеру, средняя стоимость UNIX в семидесятых годах прошлого века составляла шесть-семь тыс. долларов и была доступна далеко не всем желающим.

RSX-11M

Подсмотреть будущее — значит, украсть мистический огонь от священного костра.

Ф. Херберт. "Дюна"

В начале семидесятых еще никто не представлял себе, какой размах примет компьютеризация через десяток другой лет. Некоторые компании, пытаясь захватить ничейный рынок микрокомпьютеров, развернули в этом направлении свою деятельность. Вряд ли они догадывались о том, какой выгодной их позиция станет через несколько лет, поэтому и не прилагали особых усилий, чтобы удержаться на этом месте подольше. В то время эта ниша рынка никого не интересовала. Уж больно бесперспективным казался в то время компьютер, рассчитанный на массового покупателя.

А вот в отношении операционных систем для PDP-11 появился заметный вакуум. UNIX требовала на несколько порядков больше мощности и попросту не могла работать на компьютерах такого класса. 32 Кбайт памяти было слишком мало, чтобы разместить такого монстра.

И вот компания DEC в 1971 г. решила на создание собственной операционной системы RSX-11M. Необходимо было поддержать многозадачность, планировщик, иерархическую файловую систему, виртуальную память. На все это отводилось только 16 Кбайт оперативной памяти, так как другие 16 Кбайт было решено отдать приложениям.

Разумеется, выбор пал на ассемблер. В то время, кстати, он претерпел первое серьезное изменение. Появились директивы условного ассемблирования. Это позволило легко модифицировать исходный текст: вносить одни фрагменты, удалять другие и при этом всегда была возможность вернуться назад, если вдруг последнее изменение приводило систему к краху. Благодаря этой новой технологии лучшие специалисты DEC смогли в бешеном темпе завершить свою работу за 18 месяцев. При этом в конечном счете операционная система получилась очень простой и сильно уступающей набирающей популярность UNIX. Но своего потребителя все же нашла. Ими стали многие средние предприятия, которые могли себе позволить купить несколько ЭВМ — по одному для каждого отдела. И их вполне устраивала мощность простого 16-разрядного процессора PDP.

Вскоре RSX-11M завоевала завидную популярность, и объемы продаж были весьма внушительными. Другие фирмы, пытаясь перенять этот успех, переносили существующие приложения на мини-компьютеры, чем вызвали первую волну переориентации разработчиков. Возникал устойчивый рынок мини-компьютеров для "бедных" клиентов. В то время он еще не был таким перспективным, как сейчас, и не приносил миллиардных доходов, но требовал от разработчиков изрядного таланта — втиснуть изрядно "пожирневшее"

за последние годы программное обеспечение в скудный объем памяти мини-компьютера, сохранив при этом приемлемое быстродействие. Достаточно эффективных компиляторов в то время еще не существовало, и единственным выбором оставался ассемблер. Старым талантам вновь нашлась работа. Но теперь им пришлось бороться за рабочие места в конкуренции с молодым поколением. Неудивительно, что в новых коллективах царил совсем другая атмосфера, и... новое поколение жило по своим законам. Информатика становилась все более массовым явлением, и в нее вливалось все больше и больше случайных людей.

Среди них были и те, кто не сомневался в успехе микрокомпьютеров и делал первые шаги по созданию персонального компьютера для широких масс. Того бы, кому это удалось, ожидали бы чудовищные объемы продаж. Но для этого было необходимо, прежде всего, удешевить центральный процессор.

Intel

Солнце не просит о милосердии.

Ф. Херберт. "Дюна"

Фирма Intel была одна из первых, кто рискнул сделать ставку на микропроцессоры для персональных компьютеров (впрочем, самих терминов "персональный компьютер" и "микропроцессор" тогда еще не существовало, и их еще предстояло изобрести). Персональные компьютеры еще назывались "интеллектуальными терминалами больших машин", а микропроцессоры рассматривались исключительно как компонент управления электронными системами — светофорами или лифтами, например.

Фирма Intel не обладала значительным начальным капиталом, и на конкуренцию с существующими фирмами ей рассчитывать не приходилось. Все, что оставалось, это вложить все средства в разработку микрочипа и занять ту нишу рынка, на которую пока никто серьезно не претендовал.

В 1972 г. появился чип 8008, представляющий собой улучшенный вариант 4004 процессора и, как и его предшественник, пригодный разве что для калькулятора. Тем не менее юный Билл Гейтс все же ухитрился создать на его основе устройство для анализа трафика уличного движения, которое к "компьютерам" нельзя было отнести даже с большой натяжкой.

Но время не стояло на месте, и инженеры Intel не сидели, сложа руки. Через три года журнал "Electronics" опубликовал сообщение о новом чипе Intel 8080. Это действительно был шедевр, который каждый мог приобрести всего за 200 долларов. Конечно, этому микропроцессору было далеко до "больших машин" как по скорости, так и по архитектуре или системе команд, но для "персонального" использования он вполне годился. Потребовалось меньше года, прежде чем в мире появился первый персональный компьютер на его основе. Но что это был за компьютер! С высоты сегодняшних позиций нам

действительно очень трудно понять, что машина без клавиатуры, дисплея и конечно уж без каких бы то ни было дисковых накопителей, может называться "компьютером".

Однако это и был первый минимально функциональный ПК! Микропроцессор, немного оперативной памяти и сопрягающих микросхем — вот и все, что нужно, чтобы эта груда железа могла работать. Единственным устройством вывода служили 16 индикаторов на передней панели, а устройством ввода — 16 переключателей. Это было устройство для радиолюбителей и электронщиков. 4 Кбайт памяти тогда казалось более, чем достаточно для таких целей. А вот отсутствие удобных средств ввода/вывода сильно обескураживало. Поэтому практически каждый, купивший Альгаир, пытался в той или иной степени его модернизировать. Не будем забывать, что в то время он стоил 1000 долларов, и позволить себе его купить могли только фанатично преданные электронике (или компьютерам) люди. Теперь они получали ЭВМ в свое личное распоряжение и могли экспериментировать с ней, как им вздумается. Почти все из них были "молодым" поколением, никогда не сталкивавшимся с "серьезными" машинами. Поэтому их не смущало, что система команд и архитектура микропроцессора фирмы Intel уступала даже первым моделям PDP и требовала совсем иного мышления и подхода при работе с ней.

Разрыв в мышлении между старыми и новыми программистами был просто огромен. К тому же, если первые были большей частью образованные люди, то последние в основном кустари, безусловно талантливые, но все, за редким исключением, так и не получившие законченного образования. Но всех их объединяло одно большое чувство любви к технике и компьютерам. Не был тому исключением и Билл Гейтс, который вовсе не из-за денег в течение пяти месяцев создавал первую версию Basic для Альгаира. Это было необычайное расточительство и без того скудных машинных ресурсов, но упрощало общение с машиной. Все же Basic выучить было куда легче, чем бессмысленные (для большинства) команды ассемблера.

Впрочем, тогда эта версия Basic так и не снискала популярности. Слишком прозорливой она оказалась, и программы еле-еле ползали. Конечно, последнее мало кого могло устроить. Но джинн был выпущен из бутылки. Персональные компьютеры начали производиться многими фирмами и успешно находили своих клиентов.

Хаос

Хаос существовал в виде сырья, из которого творится порядок.

Ф. Херберт. "Дом Глав Дюны"

Перенесемся на два десятка лет назад. Многие до сих пор помнят это время. Десятки моделей машин от разных производителей, совершенно несовмес-

тимые друг с другом даже на уровне переноса текстовых файлов данных. Это стало настоящим проклятием программистов восьмидесятых годов. Обмен программным обеспечением был невозможен. Портательных компиляторов не существовало. Более того, диалекты одного и того же языка реализовывались на каждой машине по-своему.

Словом, компьютерный мир находился в состоянии полной анархии и неудержимо катился в пропасть машинной несовместимости и полной изоляции. Это порождало множество экономических и социальных проблем. Начнем с того, что преподавание в высших учебных заведениях информатики было очень затруднено невозможностью выбора одной конкретной платформы. Действительно, вероятнее всего, что в дальнейшем студент столкнется с совсем другими моделями ЭВМ иной архитектуры.

Конечно, каждый грамотный инженер должен уметь осваивать технику по предоставленной документации (да так оно и было). Но это требовало значительных затрат времени, в течение которого предприятие оплачивало обучение (а точнее переобучение) инженера. Все это негативно сказывалась на развитии рынка программного обеспечения. Фактически все программы создавались инженерами тех предприятий, которым эти программы собственно и требовались. При этом чаще написанные пакеты оказывались бесполезными для окружающих, потому что вероятнее всего последние имели совершенно другие компьютеры. Иногда совместимый компьютер находился за сотни километров в совершенно другом городе.

Конечно, каналы распространения программ все же существовали. Но чаще всего приложения, написанные "под себя", не годились для остальных пользователей. Это был "серебряный век" хакерства, особенно в нашей стране, когда требовалось перекроить чужую программку под свои нужды. Специалисты, умеющие это делать, высоко ценились в глазах руководства. Им позволялись вольности, недопустимые для простых смертных. Действительно, специалистов такого класса было немного. Хакерству не учили в учебных заведениях, и в то время не существовало никаких развитых средств обмена информацией. Нужен был изрядный талант, чтобы глубоко освоить системное программирование, как тогда было приятно говорить, без отрыва от производства. Это же можно сказать и о прикладных программах.

Однако мы забываем, что в то время существовали десятки разнотипных машин, и это обесценивало большую часть проделанной работы при переходе на новую платформу. Это было настоящим проклятием программистов. Менялся не только набор инструкций процессора и логика взаимодействия с периферией, но и сама концепция с архитектурой.

Новые модели компьютеров покупать было невыгодно. Увеличение вычислительной мощности не компенсировало необходимость переобучения персонала и переписывание всего необходимого ПО. Большинство предпочитало работать на старом оборудовании, попутно решая головолomную проблему