установка, настройка и оптимизация *nix-систем

СПЕЦ ХАКЕР

СПЕЦ ХАКЕР

№02(51) ФЕВРАЛЬ 2005

ЕЖЕМЕСЯЧНЫЙ ТЕМАТИЧЕСКИЙ КОМПЬЮТЕРНЫЙ ЖУРНАЛ

*nix без проблем

Установка, настройка и оптимизация *nix-систем

В ЖУРНАЛЕ

НА CD

Quake3 Arena 1.32 ■ IDA 4.7 FreePascal 1.9.6 (DOS/UNIX/Win32) WINE 20041201 (Src/FreeBSD/Debian) ■ Firefox 1.0 NMAP 3.75OpenOffice 1.1.4 ■ XChat 2.4.1 (UNIX/Win) XMMS 1.2.10 ■ DrWeb 4.32.2 (полный пакет)

(game)land

ISSN 1609-1027

CONTENT:

- Спец 12(49), Идеальный ПК
- Хакер 12(72)
- Железо 12(10)
- Мобильные компьютеры 12(51)
- Обновления для Windows за месяц

СПЕЦ CD

НА ДИСКЕ:

Весь софт из номера:
- FreePascal 1.9.6 (DOS/UNIX/Win32)
- WINE 20041201 (Src/FreeBSD/Debian)
- Quake3 Arena 1.32
- FreeCIV 1.14.2
- Firefox 1.0

... софт, который поможет тебе стать профи!

+ ко всему:
- Игры под *NIX
- Из *NIX в Win
- *NIX Pro
- Лучший софт от NoNaMe
- Очередной бонус от SH8

Обновления Windows (9x/XP/NT/2000/2003)
Спец 02(51), *nix без проблем
Декабрьские номера: Хакер, Железо, МС

И ЕЩЕ:

ВЕСЬ СОФТ ИЗ НОМЕРА!

ИГРЫ ПОД *NIX
- Battle Of Survival 1.1 (UNIX/Win)
- Clanbomber2 0.9
- FreeCIV 1.14.2
- Legacy 1.42
- Quake3 Arena 1.32
- Набор игр от Loki Games
- Wargus 2.1

ИЗ *NIX В WIN
- Crossover Standart 4.1
- FreePascal 1.9.6 (DOS/UNIX/Win32)
- WINE 20041201 (Src/FreeBSD/Debian)
- wxWidgets 2.4.2 (+ HTML help!)

*NIX PRO
- AFirefox 1.0
- MPlayer 1.0pre6
- NMAP 3.75
- NMAP Win 1.3.0
- OpenOffice 1.1.4
- PSI 0.9.3 (UNIX/Win)
- WGet 1.9
- XChat 2.4.1 (UNIX/Win)
- XMMS 1.2.10
- libinit 1.1.10-2
- ctrace 1.2
- evlog 1.6.1
- gdb 6.3
- totalview
- DrWeb 4.32.2 (полный пакет)
- 624 1.0.0
- ALD 0.1.7
- BView 5.62
- BurnEye 1.0.1
- IDA x86 emulator 0.5
- IDA 4.7
- Linice 2.1
- Pice 0.99
- Shiva 0.95
- Truss 0.6.7 (+ HTML помощь)
- UCL 1.03
- UPX 1.25
- Fluxbox 0.9.9
- GTK+ 2.2.4
- Xfce 4.0.6/4.1.99.3 (+ темы)

СОФТ ОТ NONAMEE
- AutoSpell Complete Check v6.2
- ACDSee PowerPack v.7.061
- MirandaIM 0.4rc1
- SIM 0.9.3
- Quick Image Viewer v1.5a
- LanScope 2.9.1
- LanSpy 1.2.1
- LanSend 1.3
- LanSafety 1.0
- LanShutDown 3.0
- LanLoad 0.9.4.1
- LanCalculator 1.0
- NetPromoter Site Statistics v2.0
- WebSite-Watcher v4.02
- Xp Slipstreamer v1.0
- URL Monitor v1.0

+ бонус от группы SH8

Root@localhost # ... вот и настал тот день, когда ты решился поставить *IX. Теперь главное - это удержаться от первого порыва и не удалить этого монстра :) Если у тебя хватило мужества, и ты-таки набрал свой первый "ls", то этот диск для тебя. А если ты испугался, то этот диск тем более для тебя, ведь "UNIX - это просто"! :)

INTRO

Давно пора признать, что Windows далеко не единственная на всем белом свете операционная система для x86-систем. Да, у нее красивый установщик, приятный и дружественный GUI, огромное колличество прикладных программ и средств разработки... Но все-таки Windows - это не единственная ОС. И не лучшая. В этом номере мы постарались как можно лучше дать всем понять, что *nix-системы - это не так страшно, как кажется на первый взгляд. Что настройка декстоп-системы - это дело пяти минут, что установить *nix можно так же, как и Windows: бестолково кликая кнопку Next. Что не только в Windows есть красивые и удобные IDE для разработчика и что отладчики в Linux - это не только gdb. Некоторые *nix-системы давно уже догнали в плане usability Windows (а по мнению многих экспертов, и обогнали). Не подумай, я вовсе не призываю тебя удалять Windows и срочно ставить FreeBSD себе на домашний компьютер, я только хочу сказать, что Linux, BSD и другие клоны Unix - это не только отличные серверные системы (в чем все уже давно убедились), а серьезный конкурент ОС от Microsoft. Будем надеяться, что эта конкуренция будет происходить не только на страницах периодики, но и в жизни тоже: в конце концов мы, пользователи, от этого только выиграем.

Gorl

СОДЕРЖАНИЕ № 02 (51)

MAKE INSTALL

DESKTOP

CODING

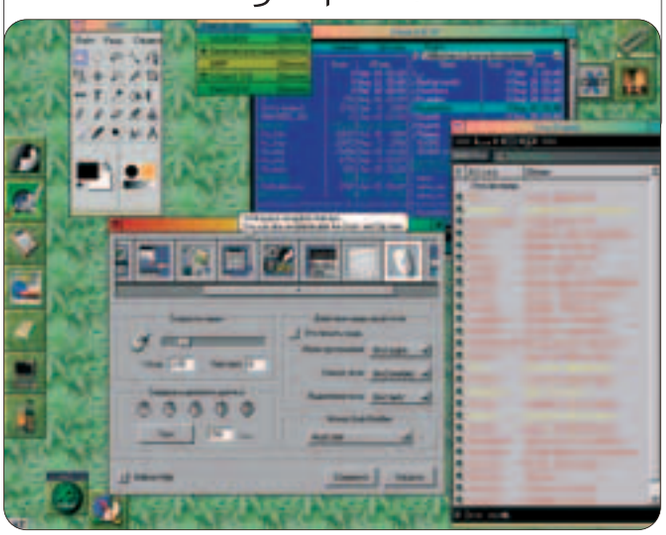
SPECail delivery

Make install

30 Граница на замке
Поднимаем безопасный и функциональный шлюз для локальной сети

Desktop

62 X-окошки
Графическая система Linux под прицелом



ОФФТОПИК

СОФТ

HARD

CREW

STORY

Desktop

66 Counter Strike под Linux
Поднятие игрового сервера

HARD

114 Маленький печатающий комбайн
Тестируем принтер Samsung ML-1520P

Редакция
» **главный редактор**
Николай «AvaLANche» Черепанов
(avalanche@real.xakep.ru)
» **выпускающие редакторы**
Ашот Оганесян
(ashot@real.xakep.ru),
Николай «Gorlum» Андреев
(gorlum@real.xakep.ru)
» **редакторы**
Александр «Dr.Klouniz» Лозовский
(alexander@real.xakep.ru),
Андрей Каролик
(andrusha@real.xakep.ru)
» **редактор CD**
Иван «SkyWriter» Касатенко
(sky@real.xakep.ru)
» **литературный редактор**
Валентина Иванова
(valyivanova1@yandex.ru)

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

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

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

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

Горячая линия по подписке
тел.: 8 (800) 200.3.999
Бесплатно для звонящих из России

Для писем
101000, Москва,
Главпочтамт, а/я 652, Хакер Спец

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

E-mail
spec@real.xakep.ru

Отпечатано в типографии «**ScanWeb**», Финляндия

Зарегистрировано в Министерстве Российской Федерации по делам печати, телерадиовещанию и средствам массовых коммуникаций **ПИ № 77-12014** от 4 марта 2002 г.

Тираж **42 000** экземпляров.
Цена договорная.

Content:

Дмитрий Шурупов (root@nixp.ru, www.nixp.ru)

МИР *NIX ЗА ПОСЛЕДНИЙ ГОД

САМЫЕ ЗНАЧИТЕЛЬНЫЕ НОВОСТИ И СОБЫТИЯ

Мир *nix-систем, а особенно открытого программного обеспечения, является одним из самых динамично развивающихся во всей компьютерной индустрии. Сообщество open-source всегда славилось постоянной активностью, и 2004 год не был исключением в этом отношении.

ЯНВАРЬ

Уже в самом начале года появляются первые сообщения о создании нового российского дистрибутива - Linux XP, основанного на Red Hat. В рунете начинается относительно активное обсуждение проекта, но долго не появляется никаких известий о продукте, и поэтому "уже сейчас лидирующий Linux-дистрибутив в России" (так написано на его официальном сайте) быстро забывают. Тем временем значительно "прославившаяся" среди сторонников open-source компания SCO продолжает доказывать свои права на Linux и рассылает вторую партию писем крупным "незаконным пользователям" открытой системы (в частности, был "отдельный разговор" с Google, подан иск на Novell). В ответ на такие действия Novell объявляет о предоставлении юридической защиты от подобных нападений своим клиентам. К этому времени окончательно завершается слияние SUSE Linux с Novell.

И корпорация Microsoft тем временем не дремлет: почувствовав угрозу своему благополучию как результат появления нового конкурента, она начинает рекламную кампанию Get the Facts, в которой доказывает, что использование Windows дешевле Linux. Компании Red Flag Software (Китай) и Miracle Linux (Япония) заявляют о появлении проекта создания собственной вариации Linux для борьбы с монополией Microsoft на местных рынках. Готовившуюся к выходу операционную систему назвали Asianux. Конец месяца ознаменовался изменением лицензии графической оболочки XFree86 и внезапным обнаружением недоступности сайта SCO (в связи с появлением вируса MyDoom, устроившего мощную DDoS-атаку на web-сервер компании). К Linux-лаборатории OSDL в январе присоединилась NEC Soft, а к группе CELF - Wind River.

ФЕВРАЛЬ

Бурно развиваются события другого громкого судебного разбирательства: Microsoft против Lindows. Еще один европейский суд (вслед за судами Финляндии и Швеции) поддерживает Microsoft и запрещает распространение продукции Lindows в Голландии, Бельгии и Люксембурге. Примечательным событием стал выпуск новой версии Linux-ядра из ветки 2.0.x-2.0.40. Mozilla, которая неоднократно выбирала не самые удачные (уже существовавшие ко времени этого выбора) названия для своего нового браузера, наконец-то окончательно разобралась с проблемой: Firebird (бывший Phoenix) переименован в Firefox. 17 февраля IBM объявляет о появлении Центра компетенции Linux в Москве. MandrakeSoft согласно решению французского суда приходится переименовать название своего дистрибутива в Mandrakelinux. Разработчики FreeBSD заявляют о приближающемся первом обновлении к 5.2-RELEASE (5.2.1) с многочисленными улучшениями в плане безопасности и стабильности и начинают долгий путь создания FreeBSD 5.3. В России ASPLinux выпускает обновление к своему дистрибутиву - ASPLinux 9.2, впервые оснащенное LiveCD-редакцией (Greenhorn). Баллмер продолжает плыть по однажды взятому курсу и во время выступления на конференции Can>Win04 в Канаде заявляет о том, что предпочтение Linux продукции его компании обычно обусловлено не экономическими, а политическими мотивами. В то же время американский штат Алабама издает билль (SB 276), объявляющий о предоставлении возможности "любому правительственному объекту использовать программное обеспечение с открытым кодом вместо платных аналогов, если такое возможно". В последний день февраля выходит XFree86 4.4.0 (с новой лицензией).

МАРТ

1 марта 2004 года выходит ОС NetBSD 1.6.2. Как водится, в ней нет ничего принципиально нового, кроме многочисленных исправлений, устранений уязвимостей и оптимизации. Уже в конце месяца разработчики начинают подготовку к выпуску NetBSD 2.0. Компания Univention и немецкое представительство SCO объявляют о заключении договоренности, согласно которой SCO прекращает предъявление претензий на код в Linux для территории Германии. Однако это не помешало последней подать иск против крупного американского поставщика автозапчастей AutoZone, использующего в своей работе Linux. 3 марта суд обязывает SCO и IBM предъявить "спорный" код в течение ближайших 45-ти дней. Mandrakesoft начинает выпуск специальных предварительных редакций Linux-дистрибутивов обнародовав релиз Mandrakelinux 10.0 Community. Молодой

Сайт кампании Get the Facts

разработчик видеокарт XGI представил родные Linux-драйверы для своей продукции. Анимационная студия Pixar решает отказаться от использования Linux (на которую она перешла с Solaris) в пользу Mac OS X. Hewlett-Packard начала продажу ПК с Linux (модели dx2000 и cd5000) в 12-ти азиатских странах. Sun недвусмысленно реагирует на призывы сообщества open-source: компания не собирается делать Java открытым проектом. Проект GNOME по причине взлома сайта откладывает релиз GNOME 2.6 до 31 марта.

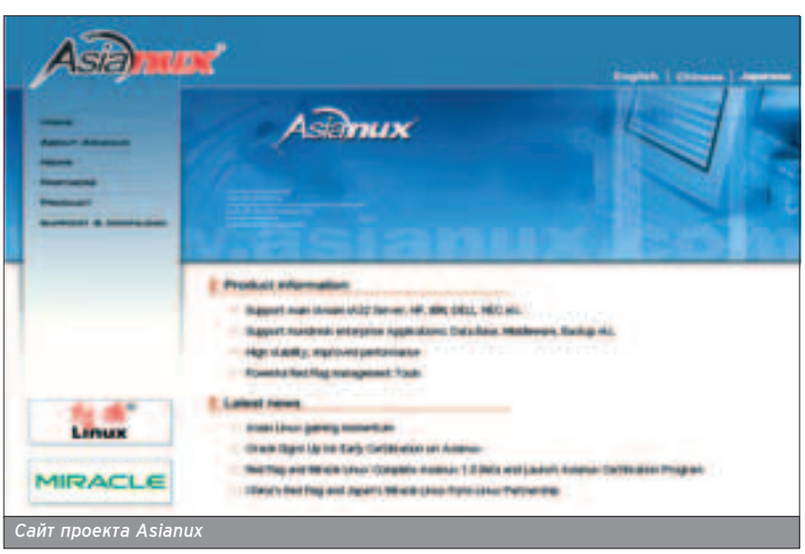
АПРЕЛЬ

■ Попытка Lindows запретить Microsoft подавать на нее судебные иски в связи с нарушением прав на торговый знак Windows провалилась, и 6 апреля Майкл Роберстон объявляет о том, что его компания и ее главный продукт (LindowsOS) получат новое название, которое стало известным 14 апреля, - Linspire. На другом фронте судебных разбирательств активизируется Red Hat: суд отказывает в просьбе SCO отклонить иск Linux-компании, а RH настоятельно просит объявить претензии SCO на Linux безосновательными и не ждать результатов дела SCO против IBM. AutoZone и DaimlerChrysler, воспользовавшись моментом, требуют отклонить обращенные против них иски SCO.

Азиаты из Red Flag Software и Miracle Linux представляют общественности первую бета-версию своей Linux - Asianux 1.0, а японская Turbolinux лицензирует у Microsoft технологию Windows Media. Корпорация Microsoft в свою очередь в начале апреля выпускает первый продукт с открытым кодом (под лицензией CPL) - Windows Installer XML (msi2xml/xml2msi), разместив его страницу на SourceForge.net. 10 апреля организация X.Org представила первый официальный релиз X Window System, появившегося как следствие изменения лицензионной политики XFree86, - X11R6.7.0, на X.Org постепенно переходят все ведущие Linux-дистрибутивы. Подразделение DreamWorks вновь рассказывает об успешном применении Linux в профессиональной анимации: 1000-процессорная Linux-ферма участвует в создании фильмов "Шрек 2" и "Подводная братва". К Linux-лаборатории OSDL присоединяется AMD.

МАЙ

■ 1 мая официально анонсируется OpenBSD 3.5, новые возможности которой представлены почти во всех аспектах работы операционной системы. Шумиха вокруг SCO немного утихает, а в компании проводят сокращение с целью "повысить прибыль от продаж UnixWare и OpenServer". Шварц, президент Sun, говорит о возможном выпуске Solaris под лицензией GPL.
3 мая 2004 года происходит занимательное событие с участием нашего соотечественника: A.I. решил проверить, насколько тщательно инспектируется добавляемый код в open-source проектах, и внедрил в код KDE комментарий на транслите. Сообщение было обнаружено разработчиками уже через полтора часа после его появления. Microsoft на SourceForge публикует второй open-source релиз (вновь под лицензией CPL) - Windows Template Library. Вашингтонский институт AdTI заявляет о том, что Linux написал не Линус Торвальдс, и что система сплошь и рядом содержит интеллектуальную собственность, "берущуюся или адаптируемую без разрешения на то со стороны владельцев материала или других компаний и индивидуумов". Тут же эти заявления критикует Эндрю Таненбаум, объясняя абсурдность подобных нападок на истинного отца Linux. По данным Gartner, рост продаж Linux-серверов за первый квартал 2004 года составил 57,3%. К Linux-лаборатории OSDL примкнул первый колледж - Marist College из Нью-Йорка.



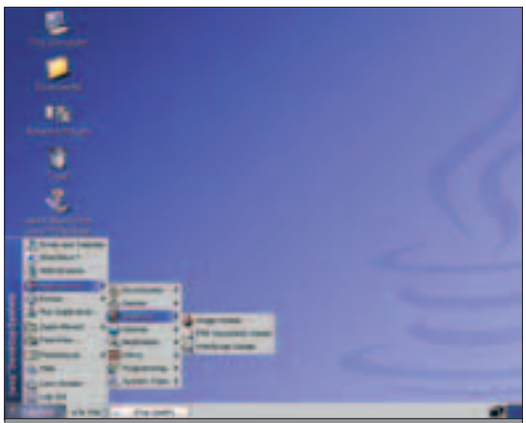
Сайт проекта Asianux

ИЮНЬ

■ Шварц подливает масла в огонь "открытой Solaris", заявляя, что Sun откроет исходные коды операционной системы, но не вдается в подробности наподобие сроков предстоящего события. SCO проявила активность и здесь: менеджер по маркетингу компании заявляет в интервью, что SCO не позволит Sun распространять Solaris под GPL. Не остановившись на достигнутом, SCO Group снова обращается в суд, чтобы добиться предоставления от IBM большего кода UNIX, чтобы уточнить, в чем были нарушены ее права. Ближе к середине июня следует сообщение о резком снижении уровня продаж Linux-лицензий от SCO, а позже компания и вовсе заявляет о том, что откладывает судебный процесс против IBM до конца 2005 года (ранее планировалось провести его в апреле 2005), анонсирует новые UNIX-продукты - UnixWare 7.1.4 и OpenServer Legend. Авторы проекта OpenBSD объявляют о том, что (по лицензионным причинам) версия web-сервера Apache, идущая в поставке с OpenBSD, остановится на релизе 1.3.29 (естественно, с постоянным выпуском исправлений). Маркетинговый директор Microsoft признает, что все больше и больше клиентов компании заинтересовываются Linux. Mozilla Foundation представляет выпуск новой версии web-браузера на базе Mozilla - Firefox 0.9. С этого момента начинается активная пропаганда готовящегося к первому финальному релизу главного будущего конкурента IE. Новая версия популярного Windows-эмулятора для Linux WineX »

Судебная активность SCO начала проявляться еще в январе 2003 года, когда компания приняла в свои ряды юриста Дэвида Боа, который и стал заниматься исследованием ситуации с патентами в UNIX.

OSDL (Open Source Development Labs) - глобальный консорциум, целью которого является повсеместное распространение ОС Linux. Именно здесь с недавних пор работает над ядром Линус Торвальдс.



Sun Java Desktop System

4.0 получила другое название - Cedega. Oracle и Red Hat сооружают Linux-центр в Сингапуре, а IBM начинает продвижение Linux в Бразилии. В последний день месяца Novell выпускает финальный релиз Mono 1.0 - открытой реализации .net.

За всю 9-летнюю историю OpenBSD в установке операционной системы "по умолчанию" была найдена всего одна уязвимость, которой можно было воспользоваться удаленно.

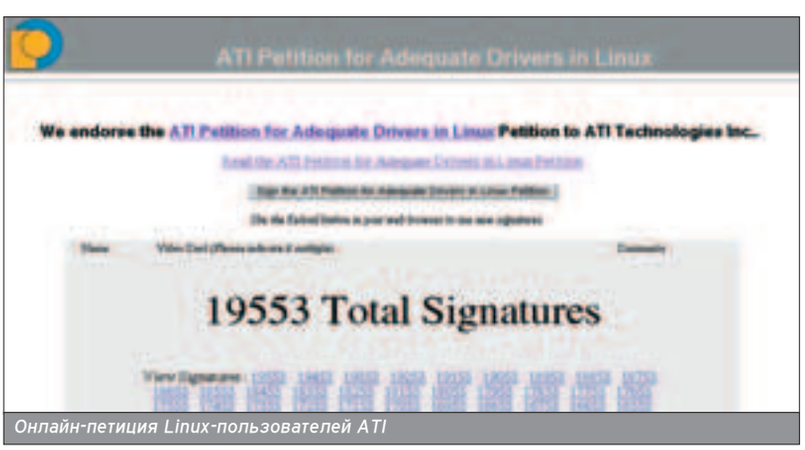
ИЮЛЬ

■ Sun сообщает об успехах JDS: ее реализацию Linux взяла на вооружение для своих компьютеров ирландская сеть банков AIB, планируя заменить ею Windows на 7500 ПК. CELF, альянс крупных производителей электроники (среди которых Sony, Panasonic, Philips), выпускает первую коллекцию открытых патчей для применения Linux в бытовой электронике. Корпорация Microsoft соглашается выплатить Lindows $20 миллионов с целью прекратить затянувшееся судебное разбирательство. Билл Гейтс во время своей поездки по странам Азии не удержался от критики open-source, заметив, что открытое ПО не гарантирует совместимости и не способствует интеграции. Несмотря на все это, правительство Южной Кореи объявило о намерении инвестировать $26 миллионов в развитие местных компаний, разрабатывающих программы для Linux. Малайзия не отстает: анонсируется план, по которому во всех правительственных поставках страны предпочтение будет отдаваться open-source. А у Red Hat падает курс акций и возникают проблемы с исками от юридических фирм, обвиняющих Linux-компанию в финансовых фальсификациях, выявленных в результате специальных проверок. Следуя примеру Linux-поставщиков, разработчики FreeBSD добавляют финальную версию патча для перехода на X.Org в -current. Новый виток развития языка программирования PHP - релиз 5.0.0 с движком Zend Engine II и значительными улучшениями (в том числе повышение функциональности). В конце июля появляется третья версия популярного интерпретатора Unix, не обновлявшегося с 2002 года, - Bash 3.0.

АВГУСТ

■ Группа OSRM (Open Source Risk Management) сообщает о том, что Linux фактически нарушила 283 патента, из которых 27 принадлежат Microsoft. RealNetworks выпускает мультимедийные плейеры Helix Player и RealPlayer 10 для Linux, причем первый поддерживает только открытые форматы (например, Ogg Vorbis). Продукцию компании поддерживают Novell, Red Hat, TurboLinux и Sun Microsystems, которые пообещали включить проигрыватели в свои Linux-дистрибутивы. Оживилась SCO Group: у компании появились новые претензии к IBM, но они уже связаны не с передачей кода Linux, а с разработкой IBM собственной версии Unix - AIX 5L. Ответ IBM не заставил себя долго ждать: компания обращается в окружной суд Солт-Лейк-Сити с требованием запретить SCO распространять любое программное обеспечение для Linux. Министерство образования Южной Кореи решает установить Linux на компьютеры, используемые в школах страны. Корпорацию Microsoft в связи с проведенной кампанией Get the Facts обвиняет в клевете на Linux британская Комиссия по рекламным стандартам, признавшая ложность "фактов", которые MS опубликовала в ходе компании. Microsoft в ответ замечает, что ее материалы были первоначально одобрены этой самой Комиссией, а сама акция уже давно завершилась, так что подобные разговоры не имеют никакого смысла.

Появляется сообщение об онлайн-петиции пользователей операционной системы Linux, требующих от ATI создать "адекватные" для их видеокарт Linux-драйверы (к тому моменту число подписчиков составило более 10000, а к концу 2004 года - около 20000).



Онлайн-петиция Linux-пользователей ATI

СЕНТЯБРЬ

■ Организация Apache Software Foundation отказывается принять Sender ID по лицензионным причинам (из-за ограничений, накладываемых Microsoft на новый стандарт), а также выпускает Java-сервер Apache Tomcat 5.5. Поддерживать Sender ID также отказывается проект Debian и несколько крупных компаний (опять же из-за лицензии). В первой половине сентября компания Sybase представляет бесплатную версию своей базы данных для Linux - Adaptive Server Enterprise (ASE), предназначенную для желающих испытать возможности корпоративной БД (2 декабря в Москве проходит семинар, посвященный ASE). Новая версия драйверов от ATI - доказательство того, что онлайн-петиция пользователей была проигнорирована. Продолжая адаптацию Linux в Бразилии, компания IBM выделила $1 миллион на создание Linux-центра, при поддержке которого со временем должны быть выпущены обучающие курсы по открытой ОС. Ближе к середине сентября появляются сообщения о портировании движка Mozilla на Qt. Mozilla Foundation тем временем выпускает предварительный релиз Firefox 1.0-PR. Проект GNOME, представив новую стабильную версию графической оболочки, решается смело заявить о том, что они "уже обогнали Windows" и стремительными темпами догоняют Apple. В середине сентября Microsoft объявляет о начале второй части рекламной акции Get the Facts, которая на этот раз будет направлена против конкретных соперников (Red Hat, Novell, IBM), а не просто против Linux.

Становятся явными первые достижения Firefox 1.0PR: за 100 часов его скачали миллион (в прямом смысле) раз, и это только начало. Microsoft, ничуть не постеснявшись своих первых двух open-source проектов, делает третий открытый релиз - FlexWiki (вновь под CPL). После более чем годового затишья в последний день сентября разработчики представили новую версию ICQ-клиента Licq - 1.3.0. Среди других программных новостей начала осени - выход клиента обмена сообщениями Gaim 1.0.0, графической оболочки AfterStep 2.0, Java 2 Platform SE 5.0.

ОКТЯБРЬ

■ Через год после объявления плана перехода с Microsoft Windows на Linux мэр г. Мюнхена (Германия) выпустил постановление о том, что, несмотря на все разногласия по осуществлению этого проекта, миграции (получившей название LiMux) быть. Шведские ис-

Сайт популяризации Firefox

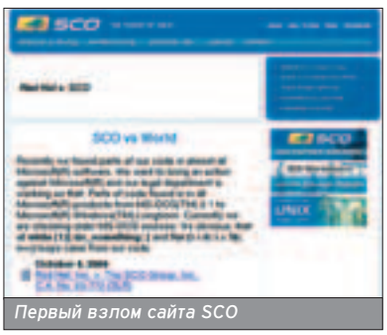
следователи ставят новый рекорд скорости передачи данных, которого добились с помощью еще не вышедшей версии ОС NetBSD 2.0. Патрик Волкердинг, главный разработчик Slackware Linux, говорит о проблемах сборки последних версий GNOME, в связи с которыми вполне вероятно его исключение из будущих релизов Slackware. И вновь Linux в школах, теперь уже в российских: в рамках программы "Дети России" в 150-ти школах Волгограда устанавливаются ПК с ОС ALT Linux 2.3 Junior, но open-source система в них не приживется, и в конце ноября все компьютеры были переведены под управление Windows. Mozilla Foundation организует акцию сбора средств на рекламную полосу Firefox в газете New York Times. И, как ни странно, добровольцев, пожелавших пожертвовать средства на благо open-source браузера нашлось не много, а очень много: нужная сумма была собрана быстрее, чем планировалось. AIX 5L от IBM стала первой (и пока единственной) разновидностью UNIX, получившей сертификацию UNIX 2003 от The Open Group, владеющей торговой маркой UNIX. В конце октября впервые от имени Red Hat проходит рассылка ложных уведомлений об обнаруженной уязвимости с предложением скачать патч, который на самом деле является трояном.

NetBSD, и без того славящаяся поддержкой многочисленных платформ, портирована на IYONIX и получает новый логотип, выбранный по результатам специально проведенного конкурса.

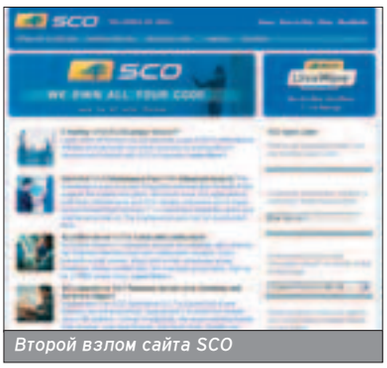
Новый логотип NetBSD

НОЯБРЬ-ДЕКАБРЬ

■ Аналитики из mi2g вновь называют Linux самой незащищенной ОС года, а исследователи из WebSideStory сообщают об очередном снижении популярности Internet Explorer (уже до 92,9%), и виновником этому, естественно, является Mozilla Firefox, чей финальный релиз выходит 9 ноября 2004 года. Также в начале ноября Sun опубликовал исходники J2SE 5.0 под лицензией Java Research License, являющейся более свободной, чем ее прошла версия. Проект FreeBSD выпускает первый стабильный релиз из пятой ветки 5.3-RELEASE, а вскоре следует выход наследницы Red Hat Linux - Fedora Core 3. Mandrakesoft объявляет о том, что наконец-то стала прибыльной. У Novell теперь тоже есть чем гордиться: во-первых, ее SUSE LINUX Professional 9.2 стала первым Linux-дистрибутивом, сертифицированным по LSB 2.0, во-вторых, финансовый отчет свидетельствует о росте доходов компании. А вот у Патрика из Slackware снова проблемы, но теперь



Первый взлом сайта SCO



Второй взлом сайта SCO

Реклама Firefox в немецкой газете

более жизненные: он болен редким заболеванием (вероятно, актиномикозом) и обращается к сообществу с просьбой помочь ему с диагностикой или лечением болезни. Linspire покупает у Microsoft лицензию на Windows Media и становится первым Linux-поставщиком в мире, обеспечившим полную поддержку форматов Windows Media 8 и 9. На наивности пользователей Red Hat вновь пытаются сыграть злую шутку: от имени компании снова поступило фальшивое сообщение об обнаружении крайне высокой степени уязвимости с приглашением скачать патч (он же - backdoor). Линус Торвальдс во второй раз выступает с обращением к Евросоюзу не принимать закон о патентах на программное обеспечение в Европе. Последние исследования IDC показывают высокий рост объемов продаж Linux-серверов: за последний квартал они выросли на 42,6% и впервые превысили уровень в $1 миллиард. В то же время доходы от продаж UNIX-серверов продолжают падать.

Доброжелательные сторонники open-source не забыли подвигов SCO: web-сайт компании был взломан как раз в день Благодарения (на месте публикаций о развитии событий борьбы SCO против Red Hat размещен издевательский текст) и еще раз после устранения последствий первого дефейса (текст на изображении многих страниц сайта был заменен на следующую надпись: "We own all your code, pay us all your money"). В последний день ноября выходит новая версия языка программирования Python - 2.4.

Немецкие пользователи Firefox успешно провели акцию информационной поддержки браузера в национальной газете Frankurter Allgemeine Zeitung: было размещено страничное рекламное объявление, в котором (в дополнение к ссылке на сайт Mozilla и к пропаганде "революционного интернет-браузера") содержится список тех, кто внес пожертвования на эту кампанию.

Сильнов Дмитрий [XL]WOLF <admin@ns0.ru>

ALMA MATER

ИЗУЧАЕМ ОСНОВЫ *NIX

Любое знакомство с новой вещью начинается с чтения инструкции к ней. Ты покупаешь мобильный телефон и ищешь его достоинства и разные функции именно в инструкции. Даже если ты знаешь, как пользоваться мобильником, то все равно есть смысл почитать эту маленькую книжицу. Эта статья задумана как инструкция по *nix-системам на примере FreeBSD.

МИР МАНУАЛА

Итак, у меня имеется в распоряжении машина: FreeBSD **.ns0.ru 4.9-STABLE FreeBSD 4.9-STABLE #7: Wed Apr 7 22:30:54 MSD 2004 xlwolf@**.ns0.ru:/usr/obj/usr/src/sys/NS i386. Есть доступ администратора. С чего начать? Конечно же, почитать инструкцию к FreeBSD. Кроме тонны книжек, которые уже сломали не одну твою полку, есть еще и электронный справочник по командам, с помощью которых ты общаешься с FreeBSD.

Конечно, хорошо бы общаться с сервером при помощи голоса, но что будет, если вместо "rm -fr ." тот услышит "rm -fr /". Лучше уж вводить команды вручную. А как узнать, что делает определенная команда (например, rm), какие у нее есть параметры и что они значат (например "-fr")? Обо всех командах и рассказано в man. Культпоход в мир Мануала начнем с команды

```
# man man
```

исполнив которую ты узнаешь, что же такое творит команда man. Теперь запусти команду man rm - и покажется мануал по утилите rm. А для того, чтобы ты понял смысл написанного, рассмотрим то, что man отображает на экране. Для отображения мануалов по умолчанию используется программа More. Итак, мануал по какой-либо команде состоит из нескольких частей:

NAME - имя самой команды, ее аналогов и краткое описание команды.

SYNOPSIS - описание синтаксиса данной команды.

DESCRIPTION - этот раздел дает подробное описание того, что делает программа и какие параметры ей можно передавать.

NOTE - здесь описаны некоторые замечания по команде. В частности, по команде rm объясняется, как можно выполнять удаление нетривиальных файлов, например, вида "-filename".

SEE ALSO - очень полезный раздел, так как тут отображаются команды, которые связаны с этой командой.

BUGS - здесь описаны известные ошибки, которые еще не исправлены.

Далее идут некоторые другие секции, которые не очень интересы, кроме одной, представляющей исторический интерес: HISTORY - здесь описывается, когда и в какой версии *nix впервые появилась данная команда.

Вроде бы все понятно, но если приглядеться, то в разделе SEE ALSO можно найти какие-то цифры рядом с командами в скобках. Зачем нужны эти неопознанные цифровые объекты? Для того чтобы все-таки опознать их, нужно посмотреть, откуда берет команда man эти самые страницы мануалов. Самое правильное - пойти в бинарник man, куда он может обращаться в пределах файловой системы. Именно это и сделаем:

```
# strings /usr/bin/man | grep "/"
```

Из всех строчек привлекает внимание запись /etc/manpath.config. После исследования этого файла становится понятно, что здесь описывается и где находятся все мануалы в системе. А основные мануалы лежат в /usr/share/man. Вот и они! И много как... man1 man2 man3... И что значат эти цифры? Мануалы структурированы по назначению команд, которые они описывают. Итак, имеем классификацию разделов:

Man1 - пользовательские команды (ls, cd, rm).

Man2 - системные вызовы (mkdir(), ioctl()).

Man3 - различные функции (printf(), sin(), abs()).

Man4 - форматы файлов (в частности, файлы в /dev).

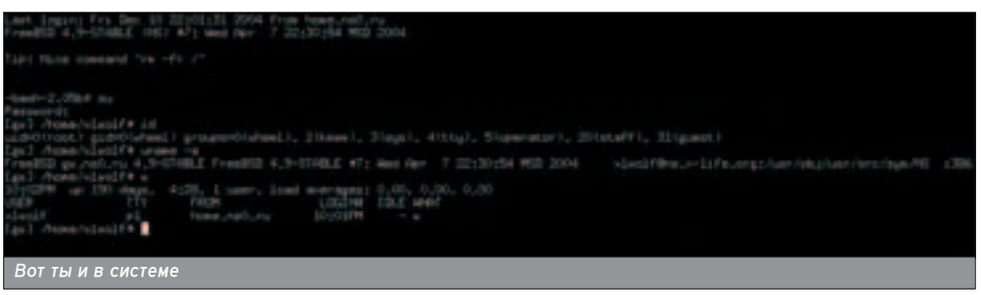
Man5 - конфигурационные файлы (hosts, syslog.conf).

Man6 - игры.

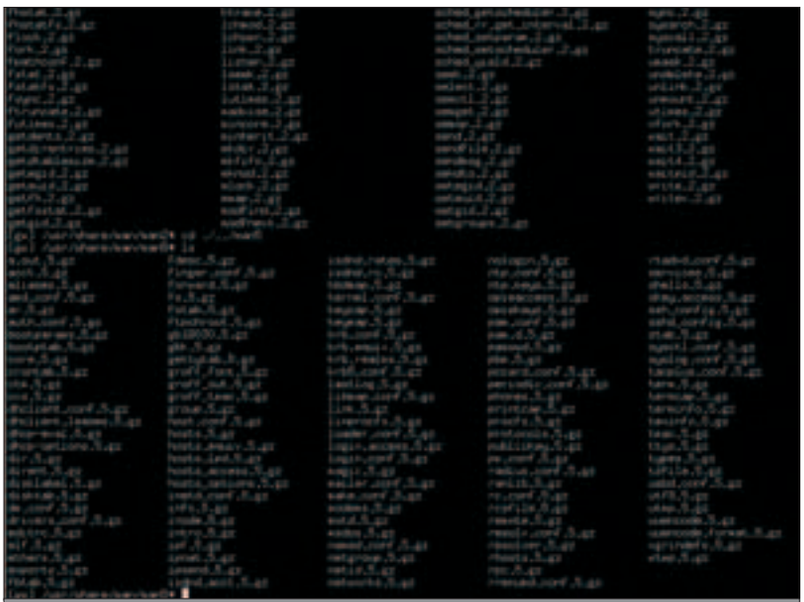
Также существуют и man7, и man8, и man9, но о них ты теперь сможешь узнать без труда сам зайдя в каталог /usr/share/man/man8.

А если вдруг мне захотелось написать свою собственную программу под *nix? Правила хорошего тона предписывают в этом случае написать мануал к ней и поместить его в нужную папку. Но и это еще не все. Так как мой мануал может занять слишком много места, то я его заархивирую. Именно так и сделано большинство мануалов. Чаще всего для этого используется gzip. Архивирование происходит при помощи очень простой команды:

Отправная точка в мир man'уала



Вот ты и в системе



Мануалы хранятся здесь

```
# gzip <имя_файла>
```

То есть если мой мануал будет называться leet_syscall.2, то написав

```
# gzip leet_syscall.2
```

я получу вместо файла leet_syscall.2 файл leet_syscall.2.gz. И размер файла уже другой. Если вдруг оказалось, что файл архивировать не нужно, то всегда можно сделать обратную операцию

```
# gzip -d leet_syscall.2.gz
```

и получить исходный файл. Также при работе с архивами часто бывает нужно, а иногда просто интересно узнать, что скрывается за личиной архива. В таком разоблачении поможет параметр "-l", то есть:

```
# gzip -l leet_syscall2.gz
```

Кроме gzip'а существует еще масса архиваторов. Рассмотрим наиболее эффективный из них – bzip2. У него совершенно особый алгоритм сжатия, и сжимает он лучше, чем gzip. Я провел следующий опыт: взял ~2 Гб текста (словари) и сжал его сначала RAR'ом, потом эти же 2 Гб - с помощью Tar, потом - bzip2. Несмотря на отсутствие у bzip2 высоких скоростей, в плане степени сжатия он оказался более эффективным, чем RAR.

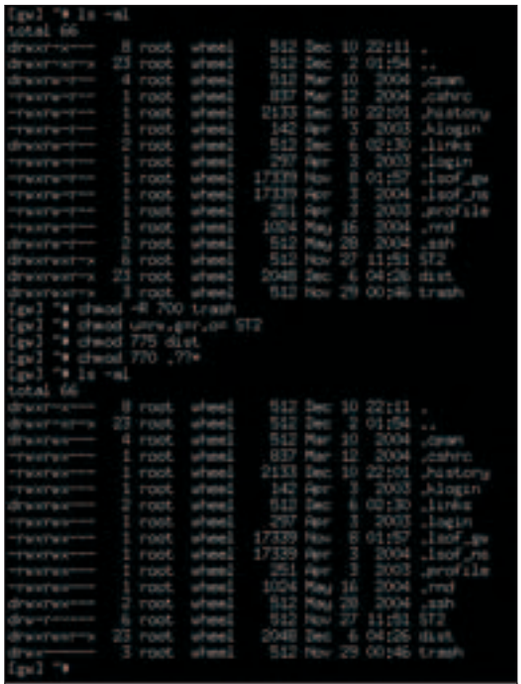
Теперь об использовании bzip2 – и снова к мануалам. Смысл работы с bzip2 такой же, как и с gzip – без параметров; сжимаешь указав только имя файла, а параметр "-d" разожмет архив. Важная особенность: ключ "-f" указывает на то, что при разархивировании необходимо перезаписывать файлы, если они уже существуют. Теперь я знаю, как экономить место на диске, но стоит уделить внимание другому аспекту моей программы и мануала к ней. Архиваторы – это, конечно, хорошо, но ведь я пишу программу только для root'а, так что я не хочу, чтобы обычные пользователи в системе могли узнать об этой программе и о том, как она работает. Но это все я описал в мануале к этой программе. Вот незадача. Но проблему опять же мне поможет решить команда man! Посмотрю-ка я мануал к команде access:

```
# man access
```

В самом верху надпись ACCESS(2). Значит так: это описание системного вызова. Это мне не подходит... Смотрю секцию SEE ALSO и вижу команду chmod(2). В ней что-то есть такое.

```
# man 2 chmod
```

Здесь двойка указывает на то, что мануал берется из секции 2 – опять системный вызов, но делать нечего: кто ищет, тот всегда найдет. Тут уже становится интересней. Доступ к файлу - это то, что мне нужно: запретить доступ к моему мануалу нежелательным пользователям! Но что-то тут совсем уж заумно написано, идем в SEE ALSO и находим команду Chmod, только уже с индексом (1). Прекрасно. Посмотри мануал к ней и пойми, наконец, что это то, что тебе нужно. Оказывается, что доступ к файлу определяется четырехзначным числом x x x x. Первый разряд определяет специальные уровни доступа, о которых расскажу позже. Второй разряд - уровень доступа хозяина файла. Третий разряд - уровень доступа для группы пользователей. Четвертый – для всех остальных. Итак, теперь ты можешь узнать, как именно определяется доступ для них. Используется очень простая и в то же время удачная система. Есть три возможности для файла (директория – это тот же файл, предназначенный для хранения других файлов) – чтение (r), запись (w), исполнение (x). Вроде бы все понятно,

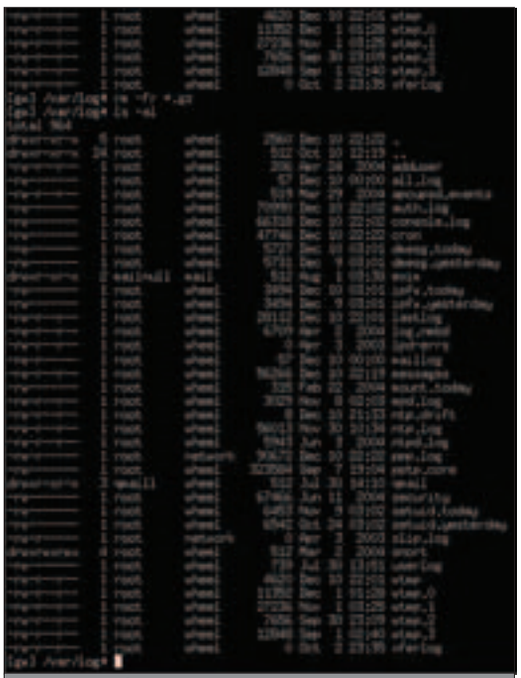


chmod - разграничит доступ.

но вот "исполнение директории" звучит как-то странно. Так и есть: смысл бита x применительно к директории имеет другой смысл: при наличии бита x можно зайти в директорию и осуществлять в ней поиск файлов.

Вернусь к моей программе и к ее мануалу. Так уж мне хочется, чтобы мой мануал мог читать только root, но, тем не менее, мою программу разрабатываю не только я, но и мой знакомый (назовем его r1c). Я и r1c, не имея доступа root'а (на данный момент я его имею, но это ненадолго), хотим в процессе разработки модифицировать как саму программу, так и мануал к ней. Для этого root'у необходимо дать право на чтение, а меня и r1c объединить в группу (devel) и разрешить этой группе чтение и запись этого файла. Вот и все. Теперь же осталось реализовать то, что было задумано. »

Мануал по bzip2



Суммируем биты доступа

Каждому биту (r, w ,x) присвоен числовой аналог. R = 4, W = 2, X = 1. Хозяином файла теперь будет root, ему нужно только чтение, у него будет доступ 4. А группе devel нужна модификация и чтение. Итого 4+2 = 6. А все остальные не должны иметь доступа к файлу, то есть 0.

Собираем все воедино: chmod 0460 <имя файла>. Это и есть результат чтения мануалов. Почему первый разряд равен нулю? О назначении этого разряда ты без труда узнаешь из мануала по команде chmod - это и будет твое домашнее задание. Иногда случается такое, что ты висишь на консоли, читаешь на досуге очередной мануал, а тут связь с сервером обрубается. Все бы ничего, но обычно это приводит с "висящим" процессам и незакрытой сессии, которая тоже зависает. В этом, конечно, нет ничего страшного, но может зависнуть и процесс ping -s 50000 www.ru, а это уже неприятно. Убей его. Допустим, из зависших процессов нужно убить именно ping. Находим его среди процессов и определяем его pid (Process IDentifier):

```
# ps ax|grep "ping"
69560 p1 S+ 0:00.01 ping -s 40000 www.ru
```

Число, находящееся в начале строки, и есть этот самый pid, в этом случае - 69560. Это уникальный идентификатор данного процесса. Зная его можно расправиться с самим процессом. Поскольку нужно убить процесс, опять идем за помощью к мануалу:

```
# man kill
```

Это и есть мануал по kill(1). С помощью этой команды можно послать сигнал процессу. Оказывается, что сигналов много и каждый из них имеет свое назначение. Я рассмотрю лишь два наиболее часто используемых сигнала - это -HUP и -9. Не странно ли, что один сигнал отображается в числовом виде, а другой в символьном? У каждого числового сигнала есть символьный аналог для удобства запоминания. Например, -1 это -HUP, а -9 это -KILL. Итак, все-таки процесс убить придется. Его pid уже неизвестен, из man видим синтаксис, поэтому:

```
# kill -9 69560
```

Я выбрал -9, потому что мне нужно обязательно завершить этот процесс, а сигнал -KILL не может быть отловлен и проигнорирован. На самом деле сигнал -HUP должен интересовать тебя больше, чем даже -9. Допустим, у нас запущен прокси squid на сервере и что-то изменено в конфигурационном файле, но работающая программа об этом ничего не знает. Не будет же она постоянно перечитывать конфигурационный файл отслеживая изменения? Можно, конечно, убить процесс и запустить squid снова, но это ниже твоего достоинства. Лучше подать процессу сигнал о том, что конфигурационный файл изменился и что его нужно прочитать. Делается это опять же просто:

```
# kill -HUP pid
```

rge pid - это идентификатор процесса, которому нужно подать сигнал. Конечно, идентификаторы - это хорошо, но не каждый человек способен сходу запомнить пятизначное число. Нужно посмотреть в таблицу процессов, найти нужный pid, скопировать или вписать его снова на консоль. Это долго, да и не всегда удобно. Тебя ждут более важные дела, а для свободы твоей гениальности придумана команда killall. Все то, что было сотворено с ping, можно было сделать проще:

```
# killall -9 ping
```

И даже не нужно было бы смотреть список процессов. Эта команда убьет все процессы ping, что не всегда желательно. Если вдруг на другой консоли сидит какой-то человек, например, мой знакомый по имени r1c, и кого-то пингует, то убийству своего собственного процесса он не порадуется. Это мокрое дело произойдет тогда, когда мы с ним будем работать от имени одного пользователя или когда я буду работать под root'ом. В том случае, если я не root и пытаюсь послать сигнал не моему процессу, мне просто будет отказано в доступе. А что сделать, чтобы не разозлить r1c'а? У команды killall есть два замечательных параметра -u и -t. Параметр -u ограничивает процессы, которым будет послан сигнал, по пользователю. То есть нужно написать:

```
# killall -u root -9 ping
```

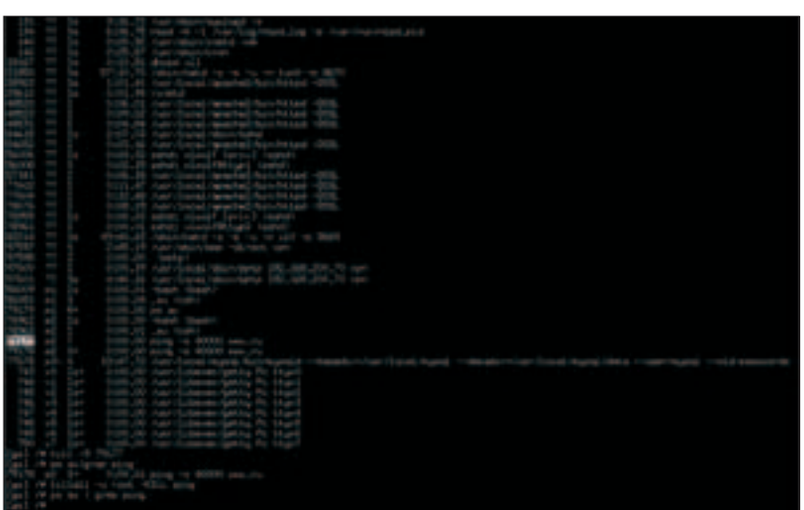
теперь будут убиты все процессы ping, которые запущены от имени root'а. Второй параметр может понадобиться, если захочешь поиздеваться над каким-либо пользователем. -t ограничивает процессы по терминалу. Что это значит и как это нам поможет? Используя команду:

```
# w
```

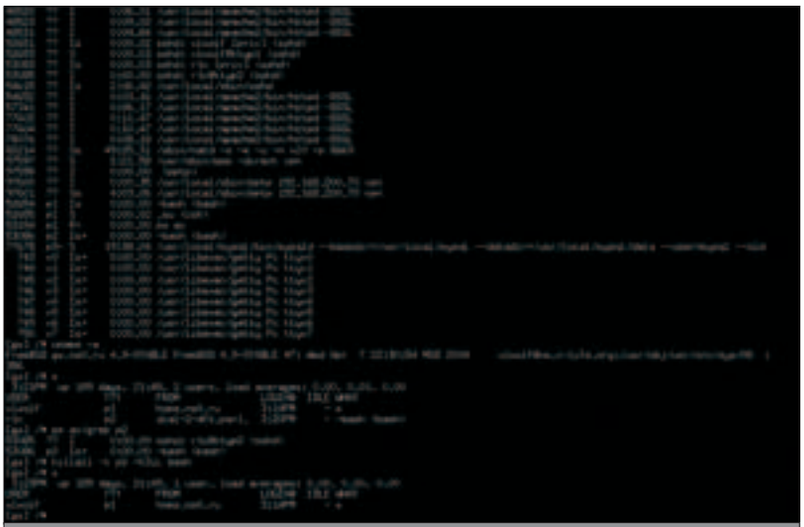
можно добраться до списка пользователей, которые работают в системе на данный момент. Итак, мы видим колонку TTY. Это и есть имя терминала, за которым работает пользователь (терминал может быть как виртуальный, так и физический; в этом случае неважно). Подмечаем, что товарищ r1c работает за терминалом p2. Выполнив команду:

```
# killall -t p2 -9 bash
```

сбрасываем r1c'а с консоли (при условии, что default-шелл у r1c'а именно bash).



Главное - убить "висячий" процесс



Лишние пользователи уходят с консоли

ФАЙЛОВАЯ СИСТЕМА

Теперь о структуре файловой системы в *nix. Можно не знать, как посылать сигналы процессам, как читать мануалы, но вот без знания устройства файловой системы, а точнее устройства каталогов, просто нельзя жить. В основе файловой системы лежит каталог с именем "/". Его называют "корневым" каталогом: он занимает самый верхний уровень в иерархии файловой системы. В нем живут все остальные каталоги системы. Итак, сделаем:

```
# ls /
```

и по порядку посмотрим, для чего нужна каждая папка.

/bin - здесь хранятся фундаментальные, основные пользовательские утилиты.

/boot - здесь хранятся программы и файлы, необходимые для загрузки системы.

/dev - здесь хранятся файлы устройств (например, замечательное псевдоустройство Urandom).

/etc - в этом каталоге можно увидеть конфигурационные файлы и скрипты. По сути, это место скопления конфигурационных файлов. Стоит особо отметить файл rc.conf, в котором определяется начальная настройка системы при загрузке.

/home - место, где обычно хранятся домашние директории пользователей (в FreeBSD 5.x это уже /usr/home) за исключением суперпользователя.

/mnt - эта папка обычно пуста, используется же она обычно как временная точка монтирования разделов.

/proc - полностью виртуальная файловая система.

/root - эта директория является домашней для пользователя root.

/sbin - здесь хранятся системные программы и утилиты администрирования.

/tmp - директория для хранения временных файлов. Не стоит доверять ей ценную информацию, потому что чаще всего администратор настраивает систему таким образом, что эта папка очищается при следующей загрузке.

/usr - основное хранилище для пользовательских утилит и программ. Тут-то и стоит искать вновь установленные программы и конфигурационные файлы к ним.

/var - здесь чаще всего хранятся логи, некоторые временные файлы, каталоги спулинга для электронной почты и принтеров и дополнительные файлы подкачки.

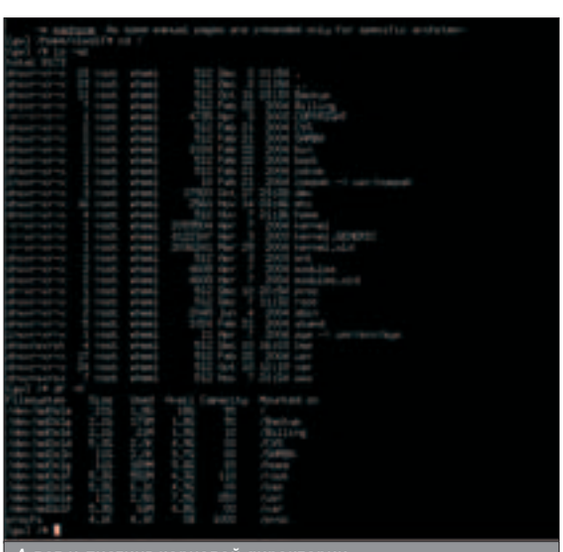
Это и есть основные каталоги в корневом каталоге. Также стоит отметить, что в каждом каталоге есть ссылка на каталог, который стоит выше по уровню. Эта ссылка имеет название "..". То есть чтобы перейти в папку, которая находится выше, пишешь

```
# cd ..
```

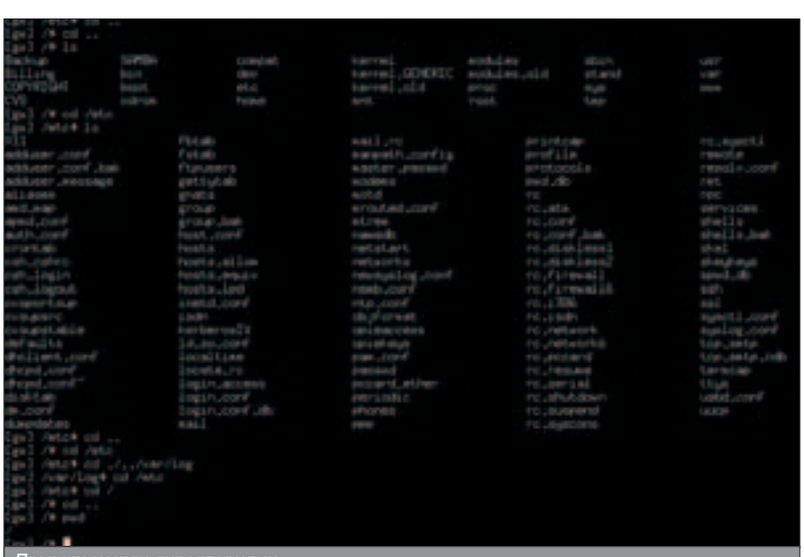
и попадаешь в директорию выше (или если ты в корневой директории - останешься в ней). Таким образом, произошло перемещение из /usr/local/etc в /usr/local.

Я попытался вложить в эту статью максимум информации, но еще больше осталось за кадром. Для того чтобы хорошо разбираться в *nix и понимать что к чему, существует лишь один рецепт успеха - истинно спецовская тяга к знанию и много-много часов, пожертвованных на чтение мануалов.

В основе файловой системы лежит каталог с именем "/".



А вот и листинг корневой директории



Прыжки между директориями

Roman AKA Docent (dOcent@rambler.ru)

СТАВЬ ПРАВИЛЬНО!

УСТАНОВКА ПРОГРАММ ПОД *NIX

Установка и удаление программ в *nix - это не намного более сложное занятие, чем в Windows, но тут имеется некоторое количество тонкостей, о которых мы и поговорим прямо сейчас. Особенно это будет полезно тому, кто только начинает общаться с *nix-системами.

В *nix-системах существует два основных способа распространения и инсталляции программного обеспечения. Первый - это стандартные bzip-, gzip- и tar-архивы, второй - rpm-пакеты. В первом случае программа после распаковки предстанет перед тобой как набор исходников (обычно на языке C/C++), который нужно компилировать при установке и указывать различные опции установки, а во втором - как бинарный самоустанавливающийся дистрибутив, не требующий компиляции и уходящий своими корнями в дистрибутив Red Hat (RPM - Red Hat Package Manager). Есть еще один вариант - это установка запуском одного файла (например install.sh или setup.sh), в котором уже прописан установочный скрипт. Такие программы или просто сразу ставятся ничего не спрашивая у тебя, или выдают какие-то свои установочные меню, или задают пользователю вопросы. Конечно, встречаются программы, не требующие установки. Поэтому прочитай прилагаемые к программе текстовые файлы, а если их нет, поищи документы на сайте разработчика или в форумах линуксоидов. Конечно, в разных программах могут быть свои тонкости вроде дополнительных опций, конфигурационных меню, но в общем и целом принцип и набор действий остается одним и тем же.

BZIP, GZIP И TAR

Это обычные архивы, аналогичные zip и RAR для Win-систем. Работать с ними в *nix можно также двумя способами. Способ первый - из командной строки. В зависимости от типа файла (gz/gz2 или bz/bz2) командами для них соответственно будут gunzip,gunzip2, bunzip, bunzip2 и название архива через пробел.

Например:

```
gunzip myfile.gz
```

Учти, что исходный архивный файл по умолчанию удаляется после распаковки, а программа распаковывается в тот же каталог, в котором лежит исходный файл, если только ты не указал какой-нибудь другой путь. Архивы tar распаковывают следующей командой:

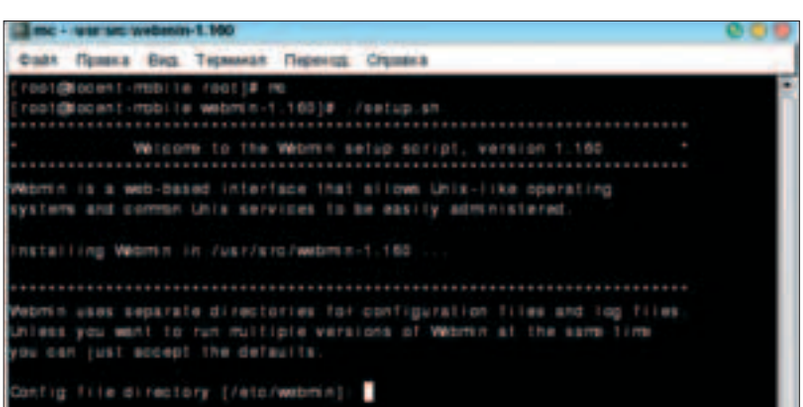
```
tar xvf myfile.tar
```

(где myfile.tar - имя файла архива). Но так как tar не сжимает файлы, а лишь упаковывает несколько файлов в один для удобства его передачи через сеть, то чаще применяется одновременное использование архиваторов BZip/GZip и tar. Файл при этом выглядит так: myfile.tar.gz или myfile.tgz. Такие файлы распаковываются следующим образом:

```
tar zxvf myfile.tar.gz
```

Вместо gunzip подставляем команду, соответствующую архиватору, которым был упакован файл. Это первый способ. Стоит отметить, что он будет актуален только если кроме консоли в твоем распоряжении ничего нет. Так было в древние времена, когда в *nix не было никакого GUI, и так делают сейчас, например, при удаленном доступе через терминал. Но знать команды *nix никогда не помешает.

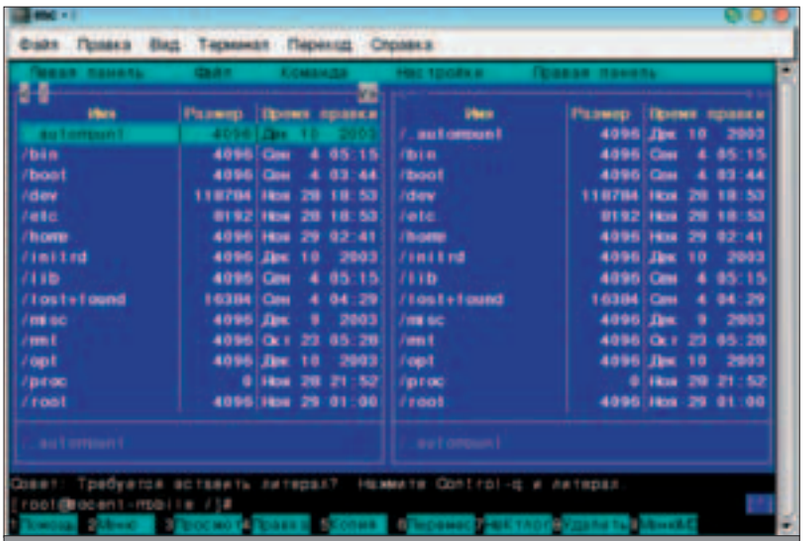
Второй способ проще. Если у тебя есть доступ к какому-нибудь файл-менеджеру, например, к знаменитому и незаменимому Midnight Commander (MC), то можно просто войти в упакованный файл, как в обычный каталог, и скопировать оттуда все содержимое в другой каталог, из которого и будешь ставить программу. При разархивировании могут возникнуть проблемы, если, например, в системе не установлен какой-нибудь архиватор, которым запакован архив. Обычно все существующие компиляторы идут вместе с дистрибутивом, но при установке программы какой-то из них может отсутствовать в системе. Или программа может оказаться запакованной какой-нибудь редкостной экзотикой. На этот случай ищи архиваторы или на установочном диске, или в интернете. Исходники программы лучше всего разархивировать в домашнюю директорию или в /tmp. Наконец, программа, а именно ее исходники, распакованы, и можно приступать к ее компиляции и установке.



Некоторые программы ставятся запуском одного файла с установочным скриптом, например, setup.sh

КОМПИЛИРОВАНИЕ ИСХОДНИКОВ И УСТАНОВКА СОФТА

Здесь все обычно идет по накатанным рельсам. Программы, поставляемые в исходниках, а следовательно, с открытым кодом, удобны тем, что их всегда можно настроить и изменить по своему вкусу вплоть до ме-



Midnight Commander существенно облегчит работу с большинством архивов

почей. Естественно, при условии, что ты разбираешься в программировании. С каждой программой поставляется свой readme-файл, который следует прочитать, так как при установке могут быть обнаружены новые тонкости. При установке программы из исходников должен быть общий алгоритм действий, который встречается чаще всего. Повторюсь: внимательно читай прилагающиеся текстовые файлы. Еще раз повторюсь: чтобы установить программу, надо обладать правами root. Итак, переходим в корневой каталог с исходниками программы, в которых обычно имеются файлы makefile и readme; набираем команду ./configure. Ждем, пока она выполнится (иногда ждать приходится долго - все зависит от объема программы); после этого набираем make, иногда make all, make config или другие "цели" (цель - это то, что указывается после make); сверься с readme для уточнения. Дальше набирай make install и жди окончания инсталляции. Программа обычно ставится по умолчанию в /usr/local/ или в /usr/X11R6/, но в установочных скриптах может быть установлен и другой путь (особенно если тебе попалось какое-нибудь обновление). Чтобы не возникло путаницы, путь можно указать вручную в самом начале:

```
./configure --prefix=путь установки
```

Сверься с readme на всякий случай. Возможно, программа по умолчанию ставится "туда, куда надо". Это общий и нехитрый набор действий для установки программы из исходников.

УСТАНОВКА ИЗ RPM

С этим все проще. В графической среде rpm пакеты должны ставится просто по клику по ним. А из командной строки это можно сделать с помощью команды:

```
rpm Uvh myfile.rpm (myfile.rpm - имя файла установочного пакета)
```

По умолчанию пакеты rpm также могут ставиться по тем же путям, что и программы в исходниках, но в них может быть прописан другой путь, предусмотренный разработчиком. При использовании rpm учти еще одну особенность: при установке информация о программе записывается в базу данных Linux, а при следующих установках или удалениях какого-либо rpm-пакета происходит проверка того, был или не был установлен раньше этот пакет, какие другие установленные пакеты от него зависят, и от каких зависит он сам. Соответственно, пакет может не устанавливаться из-за нарушений в зависимостях и не удаляться (об удалении программ - чуть позднее). Если все зависимые и зависящие пакеты установлены, а программа все равно не ставится, ругается на их отсутствие, то все равно поставить такой нужный тебе софт можно - проигнорируй предупреждения и ошибки:

```
rpm -i --nodeps myfile.rpm
```

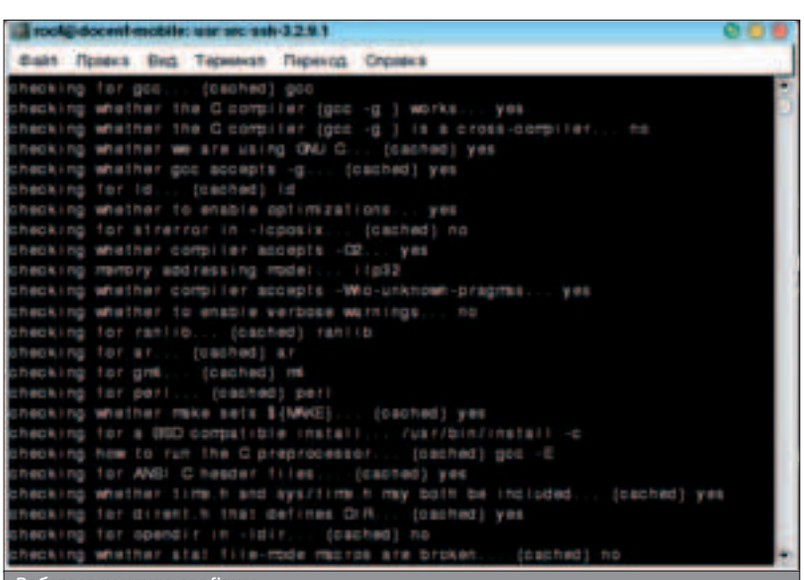
Но лучше убедиться в наличии всех необходимых пакетов и поставить их при отсутствии, иначе установленная таким образом программа откажется работать совсем или будет работать как угодно, но только не правильно.

УДАЛЕНИЕ ПРОГРАММ

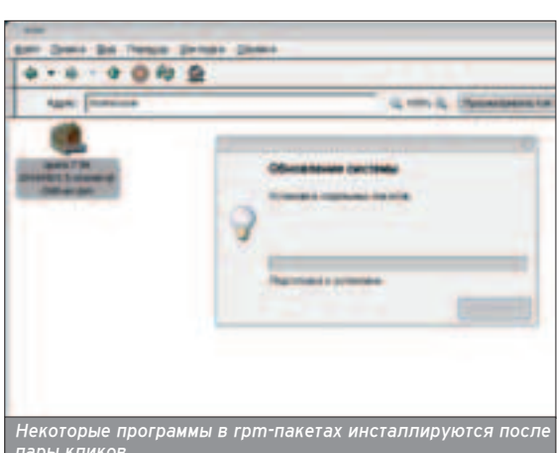
Ты наставил софта, а как теперь удалишь ненужные и занимающие лишнее место программы? Все это несложно. В случае с программным обеспечением, установленным из исходников, его можно удалить вручную (впрочем, иногда присутствует цель make deinstall): надо только точно знать, куда она устанавливалась. С rpm немного сложней, так как и здесь все зависит от пакета. Если тот пакет, который ты попытаешься удалить, связан с работой других программ, то ее удаление чревато неприятностями в системе. Rpm-пакет удаляется командой rpm -e myfile.rpm. Надо только помнить, как назывался пакет. Чтобы удалить что-нибудь обойдя предупреждения, так же, как и при установке, используй:

```
rem --nodeps -e myfile.rpm.
```

Вот и все премудрости. И никогда не забывай команду man.



Работает команда configure



Некоторые программы в rpm-пакетах инсталлируются после пары кликов

night

СТАВИМ ЧЕРТЕНКА

УЧИМСЯ ГРАМОТНО УСТАНАВЛИВАТЬ FREEBSD

Я не стану описывать все достоинства операционной системы FreeBSD. Не скажу ни слова про то, какие возможности она открывает как для тебя как администратора, так и для рядового пользователя в плане управления системой. Нарочно не упомяну о безопасности этой оси. Ты все это и сам знаешь. Я объясню, как ее установить.

Подразумевается, что у тебя уже есть диск с FreeBSD 5.3, современный i386-совместимый компьютер (он должен поддерживать диски большого размера и загрузку с CD), а также чистый винчестер либо винчестер, содержимое которого подлежит удалению (я не буду объяснять, как ставить FreeBSD второй операционной системой на машину; об этом читай в других статьях номера).

Если диска нет, то тебе придется скачать образ (644,9 Мб) с ftp://ftp.freebsd.org/pub/FreeBSD/ISO-IMAGES-i386/5.3/5.3-RELEASE-i386-disc1.iso. Для владельцев выделенки с неограниченным российским трафиком я бы посоветовал зайти на какой-нибудь ftp-поиск типа www.filesearch.ru и найти более удобное расположение файла в Сети. Скачанный образ заливай на болванку с помощью того же Nero, и будет тебе счастье.

МЫ НАЧИНАЕМ... BSD

Загружайся со скачанного или приобретенного на рынке диска FreeBSD - это первый шаг к установке BSD на твой компьютер. Через несколько секунд после старта на экране появляется чертенок, нарисованный в ASCII, и меню с вариантами загрузки. Тебя должен интересовать пункт номер один - выбирай его. Далее твоему взору откроется процесс нахождения системой различных устройств, и, наконец, ты попадаешь в меню инсталляции.

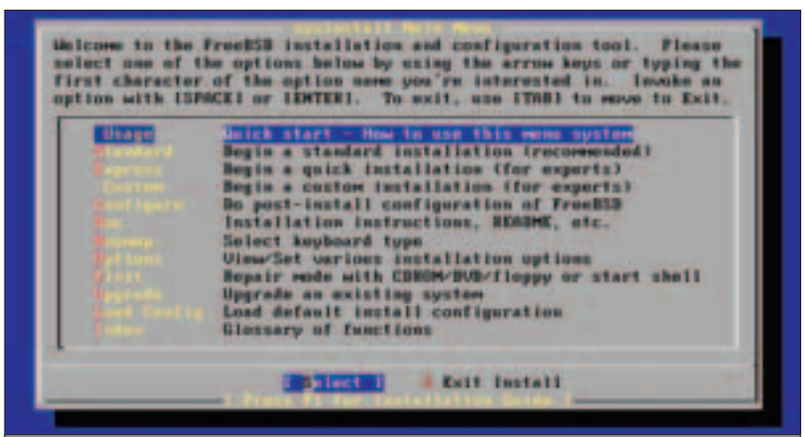
С этого момента на экране ты будешь наблюдать конфигурационную программу sysinstall (/stand/sysinstall). В ее меню предлагается выбрать метод установки - стандарт, экспресс или "выборочно". Грамотно установить систему можно только если полностью управлять процессом, поэтому выбирай третий пункт.

Стоит только нажать клавишу, как ты окажешься в меню выборочной инсталляции. Здесь тебе любезно предложат кучу самых невероятных наслаждений от настройки различных опций до разбиения диска на разделы. В опциях, по правде говоря, ловить нечего. Там можно выставить необязательные настройки при установке с сетевой файловой системы (NFS), врубить опцию ответа "ДА" на все вопросы. Если ты устанавливаешь систему с ftp, то выбирай имя пользователя. Можно отрубить warning'и и вообще врубить отладочный режим установщика. Наверняка все это действительно нужно при установке, но не в нашем случае, поэтому перейдем к разбиению диска (пункт меню №3).

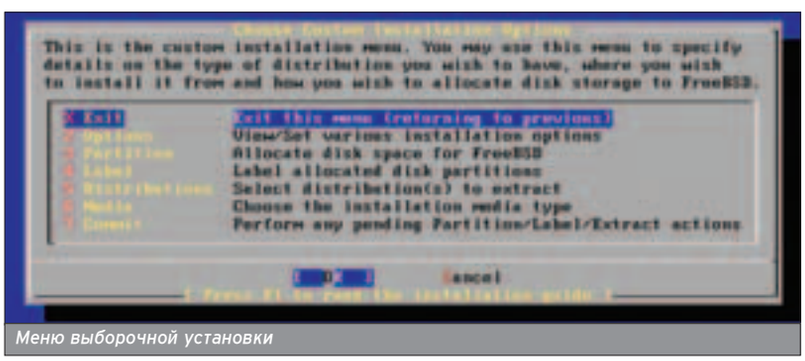
Если в компьютере два и более дисков, sysinstall предложит выбрать, на какой из них ставить систему. Если диск один-одинешенек, то установщик сразу перейдет к утилите редактирования физических разделов диска fdisk. Во FreeBSD IDE-диски обозначаются ad0, ad1 и т.д. SCSI-диски имеют обозначение da0, da1 и т.д., где ad0 - это первый физический IDE-диск в системе, первичный (primary) IDE-контроллер, master-диск. ad1 - второй IDE-диск в системе. Это может быть либо slave-диск на первичном контроллере, либо master-диск на вторичном.

В fdisk ты увидишь картину примерно как на этой картинке.

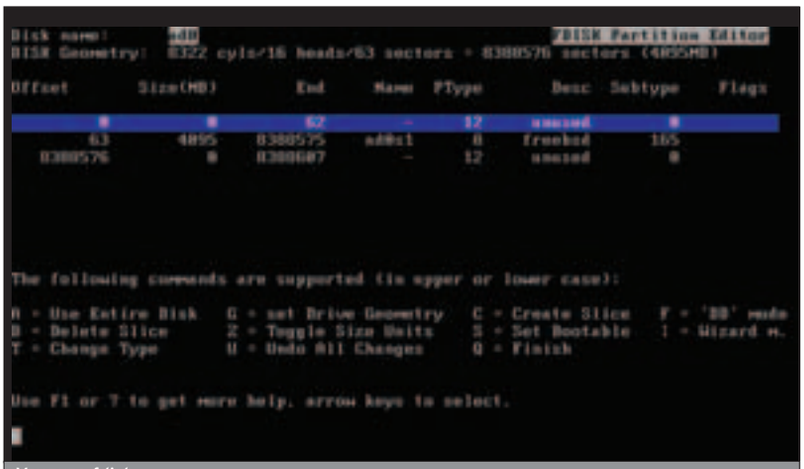
Загрузочное меню



Программа установки и конфигурации sysinstall



Меню выборочной установки



Утилита fdisk

Первая строка показывает, что мы работаем с диском ad0, вторая - информацию о геометрии диска. Далее идут данные о текущих разделах винчестера (изначально размер раздела указан в блоках и изменяется на привычные кило/мега/гигабайты нажатием клавиши "Z"). Если диск новый - будет отображен один раздел, обозначенный как unused (неиспользуемый). Если же на диске что-то есть, будь то старые разделы Windows или еще что-нибудь, то для нормальной установки тебе придется удалить все разделы кнопкой <DEL> (естественно, все данные с жесткого диска ты при этом потеряешь) и создать новый раздел. Если ты готов отвести под файловую систему BSD все доступное дисковое пространство, жми клавишу "A" (Use Entire Disk). Задать определенный размер раздела можно нажатием "C" (Create Slice), учитывая, что размер задается в блоках, а также то, что во FreeBSD разделы называются слайсами ("кусками"). Если нужно задать размер в более привычных величинах, ставь букву "M" после размера для мегабайтов или "G" для гигабайтов. Далее тебя попросят указать тип раздела. Не пугайся, тут тоже все просто: для раздела FreeBSD - 165. Нажимай и выходи, наконец, из этого дурацкого fdisk.

После выхода тебе предложат установить менеджер загрузки (BootMgr). Если на компьютере нет других операционных систем (надеюсь, все-таки их нет), выбирай в меню пункт Standard, который запретит устанавливать менеджер.

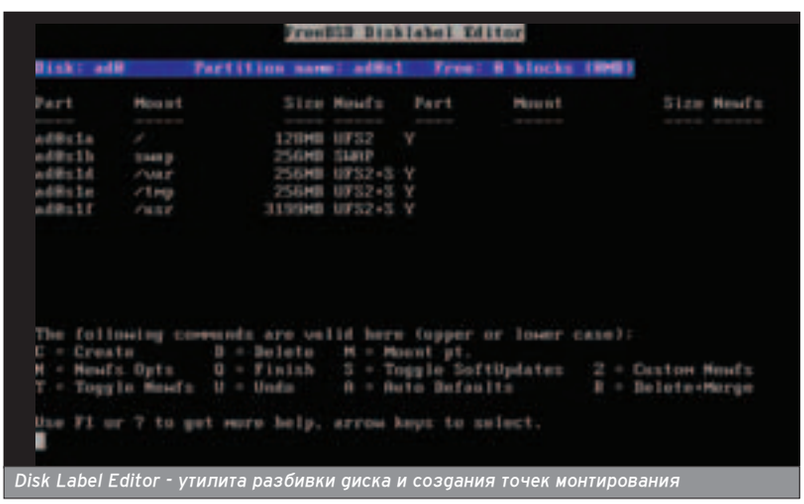
Теперь можешь смело возвращаться в меню выборочной инсталляции и выбирать следующий, четвертый пункт - Label.

ТОЧКИ МОНТИРОВАНИЯ

И попадешь ты прямиком в Disk Label Editor - программу для создания точек монтирования.

Здесь нужно определиться, будет твой компьютер использоваться в качестве сервера или это обычная рабочая станция. Если последнее - смело жми "A" (Auto Defaults), и программа сама все сделает. Но это недопустимо, если FreeBSD ставится на сервер: настройки по умолчанию тут не подойдут. Уж слишком мало места выделяется под раздел /var, в котором содержатся логи, почта и каталог спулинга для принтеров. На загруженных системах логи плодятся очень быстро, например, на хостинг-машинах гигабайт логов за ночь - обычное дело. Или какому-нибудь умнику взбредет в голову прислать по почте пару MP3'шек по 100 Мб, и останешься ты без почты и без логов...

Чтобы система работала, достаточно двух разделов - корневого и swap. Такой подход уменьшает вероятность »



Disk Label Editor - утилита разбивки диска и создания точек монтирования

переполнения одного из дополнительных разделов, но создает другие проблемы. Первая: см. выше про логи и почту плюс то, что домашние каталоги пользователей находятся в корне, а некоторые пользователи имеют свойство хранить в них гигабайты хлама. Проблема вторая: в корневой раздел будет производиться частая запись, и в случае отключения информация на нем может быть безвозвратно утеряна. Поэтому грамотные администраторы создают как минимум пять разделов, а в некоторых случаях и того больше.

Для начала создавай корневой раздел. Выделяй диск, на котором будешь создавать систему (если дисков несколько), жми "C" (Create), далее выбирай размер раздела в блоках (не забывай ставить букву M или G после числа, если указываешь размер в Мб или Гб). Для корневого раздела вполне достаточно 128 Мб.

Если же на диске что-нибудь есть, будь то старые разделы Windows или еще что-нибудь, то для нормальной установки тебе придется удалить все разделы кнопкой <DEL> и создать новый раздел.

В следующем меню выбирай тип FS (файловая система) и точку монтирования - каталог, к которому будет подключена файловая система. Вводи " / ". Проделав все эти нехитрые манипуляции, создавай swap-раздел. Снова жми "C", указывай размер и тип - Swap. Как тебе наверняка известно, размер swap'а настоятельно рекомендуется подсчитывать по формуле: 2 * объем оперативной памяти.

Как тебе наверняка известно, размер swap'а настоятельно рекомендуется подсчитывать по формуле: 2 * объем оперативной памяти.

По аналогии с созданием корневого раздела создавай остальные:

/var - здесь обосновуются, как я уже рассказывал, логи, почта и прочая канитель. Сколько выделять места под /var - решай сам, тут все зависит от того, насколько ты планируешь ее загружать. Для машины, на которой не будет почтового и web-сервера, вполне достаточно 256 Мб.

/tmp - как понятно из названия, каталог для временных файлов. Можешь расслабиться и по умолчанию оставить 256 Мб.

/usr - здесь будут ютиться почти все программы (кроме системных), отдавай этому каталогу все оставшееся место.

/home - если планируется разместить на машине хостинг или ожидаешь множества пользователей, создавай отдельный каталог /home, иначе он будет размещаться в /usr. Его объем должен зависеть от количества будущих пользователей и их наглости.

Жми "Q" и смело передвигайся на шаг вперед к установке BSD на свою машину. Следующий пункт - 5 Distributions.

ВЫБОР МЕТОДА УСТАНОВКИ

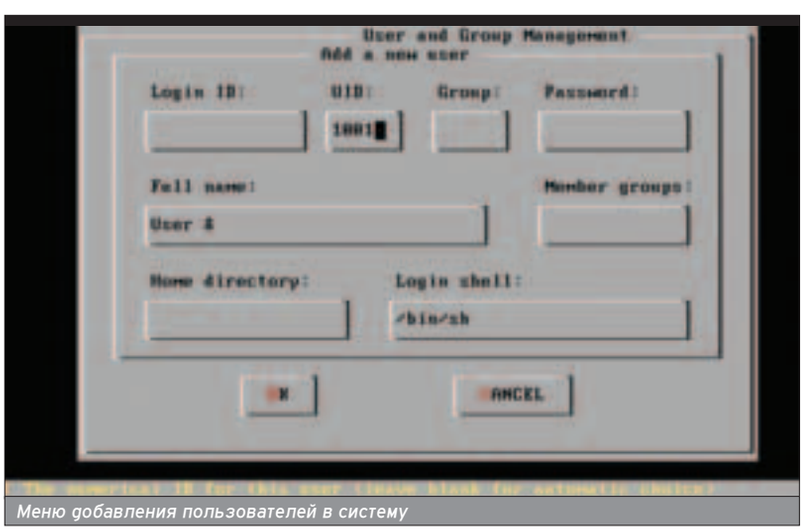
Мой тебе совет: на этом этапе установки не слишком парься и выбери какой-нибудь готовый набор (или 6 Kern-Developer, или 7 X-Kern-Developer, в зависимости от того, будешь ли ты пользоваться графической оболочкой X-Window или сидеть в консоли). Эти наборы установят весь софт (кроме пакетов), мануалы и исходники ядра. Если под /usr отведено меньше 1 Гб, стоит выбрать дистрибутивы вручную (пункт меню B Custom), не забыв также добавить исходники ядра. Без них будет невозможно конфигурировать ядро, и обновление системы через cvsup займет уйму времени.

Если ты выбрал готовый набор, тебя спросят, нужно ли устанавливать коллекцию портов. Порты - *BSD-способ установки программного обеспечения сторонних разработчиков - безусловно, вещь нужная - без дистфайлов занимает 300 Мб.

Дистрибутивы ты можешь установить со второго диска, или при твоем желании они сами скачаются автоматически при установке порта.

УСТАНОВКА

Разобрался с методами установки и дистрибутивами? Не терпится установить-таки систему? Ладно, так уж и быть. Ставь. В ответ на твое решение начать инсталлирование файлов установщик спросит тебя, откуда ставить (будто он сам не знает). Выбери CD/DVD и смело жми Commit->Yes, и начнется долгожданный процесс. Минут через 20 (в лучшем случае) инсталляция завершится, и тебя спросят, нужно ли вернуться в программу конфигурации. Ответь "ДА": мы еще не закончили.



Меню добавления пользователей в систему

ПОСТУСТАНОВОЧНЫЕ НАСТРОЙКИ

Вернувшись, ты обнаружишь, что тебе предлагают доустановить какой-нибудь софт из специальных наборов (Distributions) или из Packages. В Packages есть куча всего полезного. Разобраться, что делать, будет не очень сложно, особенно после того, как ты прошел весь этот долгий путь.

В меню Root Password тебе предстоит выбрать самый пароль администратора системы (root). Вбивай пароль, подтверждай его и сразу переходи к настройке дополнительных пользователей, так как работать в системе под root'ом считается дурным тоном. В меню User Management жми User и заполняй следующую нехитрую форму:

Login ID - имя пользователя в системе. Должно состоять из не более чем восьми алфавитно-цифровых символов в нижнем регистре.

UID - цифровой идентификатор пользователя, уникален для каждого пользователя. Нужное значение подставится автоматически.

Group - группа, членом которой будет являться пользователь. Если пользователю необходимо получать права пользователя root через команду su, добавь его в группу wheel. Также это поле можно оставить пустым, и тогда группа пользователя будет совпадать с его именем.

Password - пароль на вход в систему. Для безопасности должен содержать не менее восьми символов, в том числе буквы верхнего и нижнего регистров, а также хитрые символы вроде решетки или восклицательного знака.

Full name - реальное имя пользователя.

Member groups - в какие еще группы добавить пользователя. Если не нужно - пропускай.

Home directory - расположение домашнего каталога.

Login shell - путь к командному интерпретатору. Вводи путь к желательному для пользователя интерпретатору или оставляй как есть для использования интерпретатора sh.

После создания пользователя возвращайся в меню постинсталляционной настройки.

Пункты меню Console (настройка параметров консоли), TimeZone (настройка часового пояса), Networking (настройка сетевых адаптеров) не должны вызвать затруднений у знающих английский язык, а остальные пункты не должны вызвать интереса у трезвого человека. Выходи на самый верхний уровень меню и выбирай Exit Install.

После перезагрузки твоя система готова к использованию (хотя на самом деле еще предстоит долгий процесс конфигурации).

У НАС ОЧЕНЬ БОЛЬШОЙ ВЫБОР

* В нашем магазине вас ждет более 1000 игр на ваш выбор
* Постоянно обновляемый ассортимент
* Чем больше, тем дешевле!

Sid Meier's Pirates Limited Edition

$79,99

Star Wars Galaxies: Jump to Lightspeed

$55,99

Sims 2

$22,99

Dark Age of Camelot: Catacombs

$59,99

Half-Life 2

$23,99

Vampire: The Masquerade - Bloodlines

$79,99

World of Warcraft

$79,99

World of Warcraft 60 Day Pre-Paid Card

$59,99

Final Fantasy XI: Chains of Promathia Expansion

$55,99

EverQuest II DVD

$79,99

Need for Speed Underground 2

$22,99

Ultima Online: Samurai Empire

$59,99

Играй просто!

GamePost

ЗАБУДЬ ПРО ТЕЛЕЖКИ

МЫ ПРИВЕЗЕМ ВСЕ САМИ!

Тел.: (095) 928-0360
(095) 928-6089
(095) 928-3574
www.gamepost.ru

GAMEPOST

Антон Карпов, toxa@cterra.ru

ПЕРВЫЙ ДЕНЬ ВО FREEBSD

ТРИ ПЕРВЫХ ШАГА К КОМФОРТНОЙ РАБОТЕ

Несмотря на то, что все мы используем, как правило, одинаковый набор программ и выполняем на ПК одни и те же операции, есть некоторые действия, которые всегда полезны после установки FreeBSD. Давай взглянем на эту ОС исключительно в разрезе настольного применения, и будем помнить, что важное на сервере далеко не всегда имеет первостепенное значение для домашнего компьютера.

ШАГ №1. ПОСЛЕ ПЕРВОЙ ЗАГРУЗКИ

Прежде всего отмечу, что речь пойдет про пятую версию FreeBSD, а на данный момент актуален релиз. 5.3. 4.x уже морально устарела, и на настольном компьютере нет никаких причин использовать "четверку". Впрочем, многое из написанного мной относится и к старым релизам FreeBSD. Во время установки система предложила тебе зарегистрировать отдельного пользователя. Разумеется, ты выполнил эту операцию и не забыл добавить созданного пользователя в группу wheel, чтобы он мог повышать свои привилегии с помощью команды su. Первым делом нужно озаботиться своевременным обновлением системы. FreeBSD team, как и многие проекты, хранит исходные коды системы в репозитарии CVS - системе контроля версий, позволяющей отслеживать изменения и синхронизировать локальное дерево исходников с той или иной его версией. Подробнее про CVS и доступные ветки (branches) системы можно прочитать во FreeBSD Handbook. Ставим пакет CVSup, как наиболее удобный для синхронизации дерева исходных текстов. Причем даже если в дальнейшем планируется собирать все программы из портов, cvsup(1) проще всего поставить из прекомпилированного пакета, так как он, написанный на языке modula3, при сборке тянет в систему компилятор этого языка, и сборка затягивается.

Для работы cvsup не нужен modula3. Кроме того, cvsup имеет frontend, который абсолютно ни к чему даже на настольной машине. Итак, ставим пакет с установочного диска или с ftp-сервера:

```
~# pkg_add cvsup-without-gui-16.1h.tbz
```

После чего составим конфигурационные файлы для обновления системы, портов и документации. Они весьма просты, их синтаксис описан в том же Handbook, а рабочие примеры располагаются в /usr/share/examples/cvsup/.

```
~# cat /etc/src-supfile
*default host=cvsup5.ru.FreeBSD.org
*default base=/usr
*default prefix=/usr
*default release=cvs tag=RELENG_5
*default delete use-rel-suffix compress
src-all
~# cat /etc/ports-supfile
*default host=cvsup5.ru.FreeBSD.org
*default base=/usr
*default prefix=/usr
*default release=cvs tag=.
*default delete use-rel-suffix compress
ports-all
~# cat /etc/doc-supfile
*default host=cvsup5.ru.FreeBSD.org
*default base=/usr
*default prefix=/usr
*default release=cvs tag=.
*default delete use-rel-suffix compress
doc-all
```

Нетрудно заметить, что мы поддерживаем систему на уровне 5-STABLE, все остальное - порты и документацию - синхронизируем до текущей (CURRENT) версии. Впрочем, из-за того что применение FreeBSD на настольном компьютере позволяет экспериментировать, можно все обновлять до CURRENT, и тогда ты в качестве бонуса получишь новые возможности и новые глюки ветки 6-CURRENT :-)

```
~# cat /etc/all-supfile
*default host=cvsup5.ru.FreeBSD.org
*default base=/usr
*default prefix=/usr
*default release=cvs tag=.
*default delete use-rel-suffix compress
src-all
ports-all
doc-all
```

Возникает закономерный вопрос: "А нужно ли тянуть из портов японские, вьетнамские, еврейские и прочие локализации, и нужна ли соответствующая документация?" Конечно, нет, поэтому создаем файл /usr/sup/refuse (потому что префикс *default prefix=/usr) следующего содержания:

```
~# cat /usr/sup/refuse
doc/bn_*
doc/da_*
doc/de_*
doc/es_*
doc/el_*
doc/fr_*
doc/it_*
doc/ja_*
doc/nl_*
doc/no_*
```

FreeBSD Handbook - лучший пример сопроводительной документации

```
doc/pl_*
doc/pt_*
doc/sr_*
doc/tr_*
doc/zh_*
ports/arabic
ports/chinese
ports/french
ports/german
ports/hebrew
ports/hungarian
ports/japanese
ports/korean
ports/portuguese
ports/polish
ports/ukrainian
ports/vietnamese
```

Если ты ставил систему с компакт-диска, то установи с него же коллекцию портов и дерево исходных текстов и документации, а потом обнови их:

```
~# cvsup -L2 /etc/all-supfile
```

Можно еще больше облегчить себе жизнь внимательно прочитав /usr/src/Makefile, а потом - man make.conf. /etc/make.conf при правильной настройке позволяет здорово уменьшить количество телодвижений при обновлении системы. В данный момент нас интересуют следующие переменные, которые нужно вписать в /etc/make.conf (их названия говорят сами за себя и комментариев не требуют):

```
SUP_UPDATE=yes
SUP=/usr/local/bin/cvsup
SUPFLAGS=-L 2
SUPHOST=cvsup5.ru.freebsd.org
SUPFILE=/etc/src-supfile
PORTSSUPFILE=/etc/ports-supfile
DOCSUPFILE= /etc/doc-supfile
DOC_LANG=en_US.ISO8859-1 ru_RU.KOI8-R
```

Теперь можно обновить разом порты, исходники системы и документации одной командой:

```
~# cd /usr/src %26%26 make update
```

Обновление системы из полученных исходников отложу на потом, а пока потрачу пару минут на отключение в /etc/rc.conf лишних сервисов, совершенно не нужных на настольном ПК, а кроме этого впишу некоторые полезные переменные:

```
fsck_y_enable="YES"
usbd_enable="YES"
sendmail_enable="NONE"
sshd_enable="NO"
syslogd_flags="-ss"
```

Тем самым полностью был отключен запуск sendmail(8) и всех его агентов, syslogd'у было запрещено слушать сетевой сокет (514/udp), включен демон мониторинга USB-устройств usbd, а также была "автоматизирована" работа программы восстановления целостности файловой системы после сбоев (fsck), чтобы она не спрашивала, исправлять ошибки или нет, а молча фиксила их. Кроме этого, мы отключили sshd. Зачем он на домашней машине? Хотя если ты планируешь получать доступ к рабочей машине из локальной сети или интернета, то можешь его оставить. В остальном можно полагаться на разумные значения в /etc/defaults/rc.conf. И тут (внимание!) проявляется одна из самых частых ошибок начинающих администраторов. Отключив sendmail за ненадобностью, они забывают о том, что система все еще продолжает слать локальному root'у письма-отчеты о состоянии системы (генерируемые утилитой periodic(8). Разумеется, она пытается использовать для этого локальный почтовый сервер, который только что тихонько был убит. В итоге за пару недель/месяцев/лет эксплуатации в /var/spool/clientmqueue накапливается столько недоставленной макулатуры, что администратор узнает об этом лишь при переполнении раздела /var, когда уже поздно пить "Боржоми". Не будем повторять ошибок ушедших поколений и просто отключим системные отчеты. Все-таки это настольный компьютер. Для этого закомментируем в /etc/crontab следующие строчки:

```
# Perform daily/weekly/monthly maintenance.
#1 3 * * * root periodic daily
#15 4 * * 6 root periodic weekly
#30 5 1 * * root periodic monthly
```

Есть и другой выход: можно не убивать sendmail, а отрезать его от внешнего мира, заставив слушать только на 127.0.0.1. Все подробности этого процесса - в man sendmail или у OpenBSD, в которой sendmail по умолчанию принимает соединения только от lokalhost (/etc/mail/localhost.cf).

■ Если ты сидишь за прокси-сервером, то утилита Fetch, которая используется в системе портов для скачивания файлов, может обломиться. Дать ей указание работать через прокси-сервер ты можешь следующей строчкой в /etc/make.conf:

```
FETCH_ENV=HTTP_PROXY=my.proxy.ru:3128
```

Или, если прокси-сервер требует аутентификации:

```
FETCH_ENV=HTTP_PROXY=user:password@my.proxy.ru:3128
```

ПРИМЕРЫ ЭФФЕКТИВНОЙ РАБОТЫ С PORTUPGRADE

■ Просмотр outdated-портов, которые можно обновить:

```
~# portversion -l \%26lt;
ImageMagick %26lt;
cd2mp3 %26lt;
fluxbox-devel %26lt;
javavmwrapper %26lt;
net-snmp %26lt;
p5-BerkeleyDB %26lt;
pdflib %26lt;
razor-agents %26lt;
ru-openoffice %26lt;
Обновление outdated-портов:
~# portupgrade -arR
```

Построение индекса /usr/ports/INDEX всех доступных на текущий момент портов (требуется для работы двух приведенных выше команд):

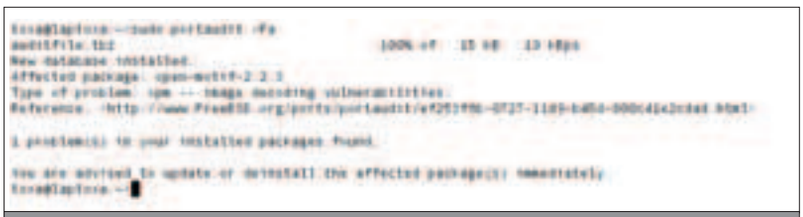
```
~# portsdb -Uu
```

Основным минусом является то, что portsdb строит индекс с нуля, что может занять много времени (портов-то более десяти тысяч). В данном случае можно посоветовать утилиту Portindex (sysutils/p5-FreeBSD-Portindex), которая генерирует INDEX инкрементально.

Венцом "первого этапа" будет обновление системы до выбранной версии, то есть сборка из исходных текстов ядра, базового окружения и документации. Но перед тем как компилировать все и вся, вспомним про волшебный /etc/make.conf. В базовое окружение FreeBSD входит много программ и сервисов, но разве нам нужен на рабочей станции сервер имен named или недавно убитый почтовый сервер sendmail? Наконец, зачем целых три пакетных фильтра (pf, ipfw2, ipf), службы UUCP, I4B (isdn for freebsd), ATM или поддержка IPv6? Тщательное прочтение man make.conf поможет сэкономить много времени при пересборке системы из исходников. Так что смело можно добавлять в /etc/make.conf как минимум следующее:

```
CFLAGS= -O2 -pipe -march=pentium4
COPTFLAGS= -O2 -pipe -march=pentium4
CPUTYPE?=pentium4
NOINET6=true
```

»



Portaudit в работе

```
NO_BIND=true
NO_SENDMAIL=true
NO_MAILWRAPPER=true
NO_KERBEROS=true
NO_LPR=true
NO_UUCP=true
NO_PF=true
NO_I4B=true
NO_IPFILTER=true
```

Разумеется, то, что уже существует в системе после установки, никуда не исчезнет, но новые версии этих программ не будут собраны, и ты можешь раз и навсегда потерять bind, sendmail, etc. Заметь, что кроме указаний на то, какие части системы собирать не планируем, мы прописали флаги оптимизации GCC под свой процессор.

Теперь нужно выполнить следующие команды:

```
~# cd /usr/src %26%26 make buildkernel %26%26
make buildworld %26%26 make installkernel
~# reboot
(система перезагрузится с новым ядром)
~# mergemaster -p
~# cd /usr/src %26%26 make installworld
~# mergemaster
```

Потом поставить "мета-порт" docproj, который установит утилиты, необходимые для сборки документации:

```
~# cd /usr/ports/textproc/docproj %26%26 make
install clean
```

А после этого соберем саму документацию. Я предпочитаю любоваться на нее в формате html:

```
~# cd /usr/doc %26%26 make FORMATS="html"
install clean
```

За подробностями всего магического процесса пересборки системы из исходников обращайся к Handbook, я же могу только констатировать, что в результате ты получишь свежую систему с необходимым набором базового софта и актуальной документацией. Неплохо для первого этапа, правда?

ШАГ №2. ДОБАВЛЯЕМ ПРОГРАММЫ

Следующий шаг - добавление в систему необходимых пакетов, без которых жизнь была бы скучна и неинтересна. Учитывая, что на десктопе, как правило, стоит современное мощное железо и что для сборки программ из исходных текстов имеется достаточно времени, то ничто не мешает использовать систему портов для добавления нужных программ.

Впрочем, в целях экономии трафика, времени и при наличии дисков с полным релизом системы первоначальную установку можно провести из прекомпилированных пакетов, а потом уже обновлять программы из портов. Первым делом поставим sudo и окончательно забудем, что такое "сидеть под root".

```
~# cd /usr/ports/security/sudo %26%26 make install
clean
```

Так как в группе wheel присутствуют только root и наш пользователь, то разрешим им выполнять команды без ввода пароля:

```
~# visudo
root ALL=(ALL) ALL
%wheel ALL=(ALL) NOPASSWD: ALL
```

Вторая по значимости программа, которую я рекомендую поставить, - это Portaudit. Этот скрипт скачивает базу уязвимых портов, сравнивает ее с установленными пакетами и показывает установленные уязвимые порты, которые следует обновить или удалить.

```
~# cd /usr/ports/security/portaudit %26%26 make
install clean
~# portaudit -Fa
auditfile.tbz 100% of 15 kB 19 kBps
New database installed.
Affected package: open-motif-2.2.3
Type of problem: xpm -- image decoding vulnerabilities.
Reference: %26lt;http://www.FreeBSD.org/ports/por-
taudit/ef253f8b-0727-11d9-b45d-
000c41e2cdad.html%26gt;
1 problem(s) in your installed packages found.
You are advised to update or deinstall the affected
package(s) immediately.
```

Если ты решил не убивать sendmail и оставить периодическую генерацию системных отчетов, то можно прописать запуск portaudit в конфигурационный файл /etc/periodic.conf уже знакомой нам утилиты periodic(8) и получать по почте отчеты об уязвимостях в установленных пакетах:

```
~# echo daily_status_security_portaudit_enable=YES
%26gt;%26gt; /etc/periodic.conf
```

Уж раз решено всецело положиться на сборку программ из портов, то без утилиты Portupgrade не обойтись. Стандартных pkg_* утилит явно недостаточно для эффективной работы с портами.

В рабочей системе может быть установлено свыше 300 пакетов

freshports.org - здесь можно узнать о новых портированных программах

А НУЖЕН ЛИ ФАЙРВОЛ?

Ты заметил, что на рабочей станции мы не особо-то уделили внимание пакетному фильтру, даже оставили "за бортом" PF и IPF. Действительно, а зачем? FreeBSD - не Windows, и светить открытыми портами во все стороны привычки не имеет. То немногое, что запускалось при старте системы, было отключено. Но если паранойя не дает тебе спать, можно ограничиться старым добрым ipfw2:

```
firewall_enable="YES"
firewall_quiet="YES"
firewall_type="client"
```

где тип файрвола - client - указан в /etc/rc.firewall. Тебе нужно лишь слегка подправить его с тем расчетом, чтобы он пропускал все соединения от тебя сохраняя сеанс (statefull filtering), но запрещал все входящие пакеты. Если твоя рабочая freebsd-станция является одновременно и шлюзом для домашней сетки, могу тебе только посочувствовать (для этих целей должна быть выделена отдельная машина :) и предложить почитать OpenBSD PF User's Guide (www.openbsd.org/faq/pf/index.html) - самый продвинутый на сегодняшний день пакетный фильтр, входящий в базовое окружение FreeBSD. Разумеется, про NO_PF=true в /etc/make.conf в этом случае лучше забыть.

```
~# cd /usr/ports/sysutils/portupgrade %26%26 make install clean
```

Вообще, в портах утилит для работы с ... портами :) очень много, но для начала достаточно Portupgrade.

Прежде чем ставить остальные программы, снова направляем взгляд на многострадальный make.conf. В нем, помимо указания системных переменных, можно указывать переменные для конкретного порта. Для этого нужно обрамить переменные в условия:

```
.if ${.CURDIR:N*/ports/editors/vim} == ""
NO_GUI=yes
.endif
```

Эта запись означает, что если считывающий ее Makefile находится в заданном каталоге CURDIR, то указанная переменная считывается как переменная окружения и используется при сборке порта. Так как /etc/make.conf считывается при сборке каждого порта, то указание опций в make.conf аналогично тому, как если бы они были указаны при компиляции:

```
~# cd /usr/ports/editors/vim %26%26 make NO_GUI=yes install clean (первый вариант)
~# cd /usr/ports/editors/vim %26%26 make -DNO_GUI install clean (второй вариант)
```

```
toxa@laptoxa:~>portversion -l \*
ImageMagick                    <
cd2mp3                         <
fluxbox-devel                  <
javavmwrapper                  <
net-snmp                       <
p5-BerkeleyDB                  <
pdflib                         <
razor-agents                   <
ru-openoffice                  <
toxa@laptoxa:~>sudo portupgrade -rR cd2mp3-0.82.1
---> Upgrading 'cd2mp3-0.82.1' to 'cd2mp3-0.82_1.1' (audio/cd2mp3)
---> Building '/usr/ports/audio/cd2mp3'
===> Cleaning for dagrab-0.3.5_1
===> Cleaning for lame-3.96.1
===> Cleaning for libiconv-1.9.2_1
===> Cleaning for gettext-0.13.1_1
===> Cleaning for gmake-3.80_2
===> Cleaning for libtool-1.5.10
===> Cleaning for nasm-0.98.38_1.1
===> Cleaning for cd2mp3-0.82_1.1
===> Extracting for cd2mp3-0.82_1.1
=> Checksum OK for cd2mp3-0.82.tar.gz.
===> Patching for cd2mp3-0.82_1.1
===> Configuring for cd2mp3-0.82_1.1
```

Обновляемся с Portupgrade - просто и удобно

Однако при обновлении порта той же утилитой portupgrade была бы "потеряна" опция, если бы ее не указали в make.conf. Теперь же не о чем беспокоиться, до тех пор пока такая опция есть в Makefile порта.

Следующая на очереди программа - это рабочий shell. Действительно, можно довольствоваться стандартным tcsh, но я предпочитаю что-нибудь более удобное. И хотя самой популярной оболочкой является bash, по возможностям сейчас нет равных Z-shell. Так что я попытаюсь обратить тебя в свою веру ;-).

```
~# cd /usr/ports/shells/zsh %26%26 make install clean
~# chsh -s zsh user
```

ШАГ №3. ПОСЛЕСЛОВИЕ

В принципе, основные настройки сделаны, теперь система может быть полностью отдана в твое распоряжение. Можешь делать с ней что хочешь и ставить что хочешь, благо 12 с лишним тысяч портов не дадут тебе скучать. Однако напоследок - пара маленьких приемов, которые могут пригодиться. Во-первых, на хороших LCD-мониторах стандартная 80x25 консоль смотрится весьма печально. Поправить это можно следующими опциями в /etc/rc.conf:

```
allscreens_flags="-g 8x14 VGA_80x30 green black"
font8x14="ter-k14n"
font8x16="ter-k16n"
```

где "ter-*" это шрифты из набора terminus, который можно взять по адресу www.is-vn.bg/hamster/terminus-font-4.11.tar.gz. Распаковав архив и перейдя в целевой каталог, следует набрать make raw для создания шрифтов под freebsd'шный консольный драйвер syscons. Затем нужные шрифты просто скопировать в /usr/share/syscons/fonts. Твоему монитору сразу полегчает :). Во-вторых, в качестве X-сервера ты наверняка используешь X.Org: он заменил опальный XFree86. Однако старый сервер все еще присутствует в системе, и чтобы при сборке портов система правильно определяла требуемые иксовые библиотеки, можно явно прописать использование xorg:

```
~# echo X_WINDOW_SYSTEM=xorg %26gt;%26gt; /etc/make.conf
```

В-третьих, если в качестве рабочей станции ты используешь ноутбук, обязательно прочти серию заметок "Мобильные Юниксы" (3 части), которые публиковались в разных выпусках X и уже доступны на нашем сайте.

На этом начальную настройку прошу считать оконченной. И помни, что на все твои вопросы ответит всемогущий Google.

Докучаев Дмитрий aka Forb (forb@real.xakep.ru)

ПРИРУЧАЕМ ПИНГВИНА

ГРАМОТНАЯ УСТАНОВКА LINUX

И так, ты решил установить Linux. Поздравляю! Некоторые личности считают, что для инсталляции пингвина особого ума не надо. Достаточно достать новый дистрибутив и настойчиво кликать Next на каждой вкладке установщика. Это не совсем так. Действуя вышеописанным методом, ты соберешь убогую неоптимизированную систему. Для максимальной же производительности нужно принимать правильное решение на каждом этапе установки.

МИНИМУМ УДОБСТВА, МАКСИМУМ ГЕМОРРОЯ

Microsoft всегда старалась привлечь внимание пользователя к своим продуктам, в том числе - красивым и простым установщиком. Поэтому чтобы установить Windows - действительно достаточно нажимать "Далее"\Next на каждой вкладке инсталлятора. С Linux все иначе. Начнем с того, что тебе нужно позаботиться о правильной разбивке разделов. Затем внимательно выбрать только нужные программы из списка. И, наконец, грамотно обдумывать свои действия на каждом шаге инсталляции.

Но, как говорится, не так страшен черт, как его трезубец. Если хотя бы одним ухом прислушаться к моим советам, система соберется без приключений и долгие годы будет радовать стабильностью и высокой производительностью (по-моему, это именно то, чего так не хватает Windows). Сейчас на твоих глазах будет грамотно установлен популярный Linux-дистрибутив с известным тебе именем RedHat 9.0.

В ДОБРЫЙ ПУТЬ!

Прежде чем грузиться с установочного CD, необходимо создать условия для установки Linux. Многие пропускают этот шаг, а затем трут раздел Windows на стадии разбивки дисков, получая при этом печальный опыт установки пингвина. Мы же не будем лишний раз наступать на грабли, а воспользуемся услугами программы Partition Magic, лучшей программы для управления дисками в Windows. В ней необходимо выбрать диск, на который ты собираешься ставить пингвина, и осуществить операцию Resize Partition. В результате ты должен высвободить объем, который будет использоваться для Linux. Естественно, что если ты собираешься ставить систему на чистый винчестер, никаких предварительных операций выполнять не требуется.

Теперь можно вставлять в привод компакт-диск с дистрибутивом и загружаться с него. После обращения к CD ты сразу же увидишь текстовое меню. Здесь можно не заморачиваться и сразу нажимать <Enter>. Это меню позволяет загрузиться с диска в том случае, если по какой-то причине пингвин откинул ласты (аналог safe-mode в Windows). Кроме того, можно запустить установщик в убогом текстовом режиме, но я советую устанавливать систему в графическом инсталляторе.

С того самого момента, когда был нажат <Enter>, началась установка Linux. Прислушивайся к каждому моему совету и будь предельно внимателен. Поехали!

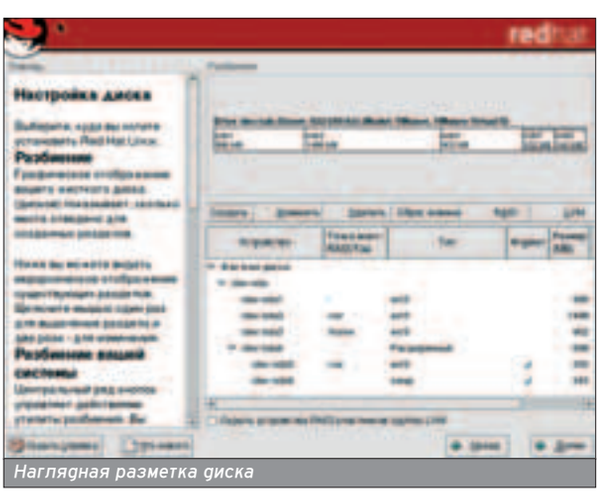
ПЕРВОНАЧАЛЬНАЯ НАСТРОЙКА УСТРОЙСТВ

Первое, что тебе предложат сделать, - протестировать читабельность установочных дисков. Если диск попал тебе не из первых рук, очень советую обратить особое внимание на этот пункт. В случае же, когда диск практически новый и на нем нет видимых царапин, можно пропустить эту длительную процедуру.

После загрузки установщика ты увидишь, что установщик приобрел симпатичный оконный вид. Перед тобой предстанет окно с поздравлениями и благодарностями за то, что ты выбрал именно RedHat. Можешь особо не вчитываться - там все равно нет ничего полезного. В следующей вкладке тебе предстоит выбрать язык установщика. Тут, по-моему, решение однозначное - русский. Сразу же после выбора языка все англоязычные надписи сменятся на родные буквы. Уже неплохо, можно отложить словарик в сторону и попытаться выбрать правильную раскладку клавиатуры. Как видишь, RedHat поддерживает как минимум две советских раскладки - Russian (KOI8-R) и Russian (cp1251). Для совместимости предлагаю выб-

Если по какой-то причине ты не хочешь устанавливать систему в графическом режиме, набери опцию Text сразу после загрузки с CD.

После установки обязательно создай загрузочную дискету. Она не раз тебя выручит, если ты напортачишь с ядром или с опциями загрузчика :).



Наглядная разметка диска

Параметры загрузчика

рать второй вариант, так как в этом случае твоя система будет полностью совместима с Windows. Если же ты предпочтешь KOI8-R, то ни капли не прогадаешь - система все равно останется русской.

Дальше показывается мастер выбора мышки. Если у тебя стандартный манипулятор, то инсталлятор сам определит его тип и достаточно лишь подтвердить правильный выбор кликом по кнопке "Далее"\Next. В противном случае выбирай Generic, и установщик проставит для твоей мыши стандартные настройки. Для несчастных обладателей двухкнопочных мышек есть пункт "Эмулировать три кнопки". При его активации нажатие третьей кнопки будет отслеживаться по одновременному клику по двум клавишам устройства.

Теперь мастер спросит у тебя тип компьютера, на который устанавливается система. Это может быть обычный персональный компьютер, рабочая станция, сервер или выборочная установка. Я настойчиво советую выбрать именно четвертый вариант, так как в остальных случаях глупый мастер соберет крайне неоптимизированную систему с огромным количеством лишних программ, которые тебе никогда бы не пригодились.

ВОЙНА С РАЗДЕЛАМИ

■ Наконец-то мы дошли до момента, когда установщик спрашивает о создании новых разделов. Никогда не доверяй автоматике и выбирай только ручную разбивку. С одной стороны, никто не исключает потерю всех данных, а с другой - только ручное создание разделов поможет сделать систему максимально удобной.

После запуска программы Disk Druid ты увидишь окно с разметкой твоего HDD и нижележащим меню операций. Кликай мышкой по свободной области на винчестере и выбирай пункт меню "Создать". В качестве типа системы указывай ext3, точка монтирования - корневая ("/"), размер будет зависеть от суммарного места, выделенного под Linux. Если, например, ты высвободил 5 Гб, то под корень можно отдать 1 (мое личное мнение).

Затем следует создать еще несколько разделов с точками монтирования /usr (базовые программы), /home (домашние каталоги) и /usr/local (программы, собранные из исходников). Максимальный размер старайся выделить под /usr и /usr/local. Что касается домашних каталогов - тут все зависит от того, кто будет прописан на твоей будущей машине. Если ты собираешься пустить на сервер друга-варезника, то позаботься о соответствующем размере раздела. В остальных случаях, думаю, хватит полгигабайта (опять же, относительно общего 5-гигабайтного объема).

И наконец, пришло время для рождения последнего раздела - swap`а. Несмотря на большие объемы оперативки, swap все-таки приходится использовать. Его размер рассчитывается по известной тебе формуле: количество оперативной памяти * 2.

В итоге все свободное пространство должно быть занято под разделы. Убедись, что на экране изображено что-то похожее на скриншот и со спокойной душой переходи к следующему этапу установки нажатием кнопки "Далее"\Next. Мастер может выругаться, что размер раздела слишком маленький, но в этом нет ничего страшного.

ПАРАМЕТРЫ СИСТЕМЫ

■ В следующем окне будут указаны параметры загрузчика. По умолчанию устанавливается grub. Помимо этого лоадера существует LILO - загрузчик, который был популярен несколько лет назад. Несмотря на урезанные возможности, разработчики не забыли включить его в список. Однако возможности GRUB в несколько раз больше LILO'шных, поэтому в этой вкладке ничего менять не нужно.

Самое главное - убедись (если ты устанавливаешь систему на диск второй), что другая операционная система, например, Windows, присутствует в списке загружаемых систем. Обычно мастер быстро распознает все ОС, которые установлены на носителе. В самом низу можно увидеть предложение установить пароль на загрузку системы. Делать это или нет - решать тебе, но если компьютер стоит дома и кроме тебя им никто не пользуется - имеешь полное право пропустить этот шаг. В случае если в твоей квартире обитает брат (сестра, отец, дядя), которого хлебом не корми - дай посидеть за компьютером, обязательно поставь пароль.

Следующий этап - настройка Сети. Если ты используешь dialup для выхода в интернет, этот этап ты можешь пропустить, так как здесь будет настраиваться только LAN. Если же локальная сеть для тебя как воздух и без ее настройки ты не проживешь и часа - вбивай свои реквизиты (IP-адрес, маску подсети, шлюз, DNS-сервер) в соответствующие поля.

Чуть дальше можно увидеть конфигурацию файрвола iptables. Здесь все аналогично брандмауэру в WinXP. Настройка ведется по принципу "кликнул и забыл" :). Включи средний уровень безопасности (самый высокий уровень бывает нужен крайне редко) и отметь галочками порты, на которые будут разрешены внешние подключения.

После определения часового пояса (думаю, свое местоположение ты выберешь без моих подсказок) тебе предложат ввести root'овый пароль. Ты наверняка знаешь, что root - это пользователь с администраторскими правами (наподобие аккаунта "Администратор" в Windows), соответственно password на эту учетную запись должен быть максимально сложным.

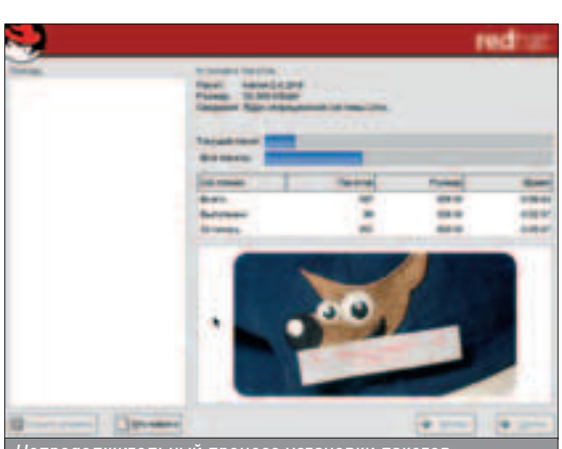
Наконец, в следующей вкладке начинается самое интересное - выбор приложений. Чтобы инсталлятор не поставил никому не нужные пакеты, отметь галочкой "Индивидуальный выбор пакетов" и переходи к следующему окну. Далее кликай по кнопке "Простой просмотр". Здесь тебе нужно медленно пройти по всему списку пакетов и отобрать для себя только нужные программы. Это делается опять же для того, чтобы получить максимальную отдачу от системы. Ты можешь спросить: "А какие программы мне нужны?" Отвечаю: для каждой проги существует свое описание на русском языке. После его чтения ты сразу поймешь, нужно ли тебе устанавливать приложение. К примеру, в описании сказано, что устанавливаемый пакет - это португальский словарь к переводчику. Избавившись от него (если ты, конечно, не ярый поклонник Portu), ты освободишь ценные мегабайты.

Теперь, как ЛЮБИТ ГОВОРИТЬ УСТАНОВЩИК WINDOWS, можешь откинуться на спинку стула и наслаждаться процессом установки... Linux. В зависимости от мощности твоего компьютера и объема пакетов, процесс займет от 20-ти минут до нескольких часов твоего драгоценного времени. По мере необходимости установщик будет просить вставить другие установочные диски (всего их три).

После того как все бинарники успешно установились, инсталлятор поздравит тебя с успешной сборкой RedHat. Мысленно поблагодари установщика и перезагружайся. Если все мои советы выполнены правильно, ты увидишь стандартное приглашение RedHat для ввода логина. Но не обольщайся. Установив Linux, ты практически ничего не сделал. Теперь твою систему нужно оптимизировать, устанавливать обновленные сервисы, настраивать файрвол. В нелегкой настройке пингвина тебе, конечно же, помогут избранные статьи этого номера.

Надо сказать, что RedHat-подобная установка встречается и у других дистрибутивов. Например, процесс установки Mandrake чем то похож на инсталляцию "красной шапки".

Помимо ext3, ты можешь выбрать и другие файловые системы (например, reiserfs или ext2). Про производительность файловых систем ты можешь также прочитать в этом номере.



Непродолжительный процесс установки пакетов

Денис Колисниченко, dhsilabs@mail.ru

ЖЕЛЕЗНАЯ СТОРОНА LINUX

УСТАНОВКА И НАСТРОЙКА ОБОРУДОВАНИЯ В LINUX

Установка нового оборудования в операционной системе Linux во многом отличается от той же процедуры в более привычной большинству пользователей Microsoft Windows, и именно от нее во многом зависит работоспособность и производительность твоего компьютера. Подробнее об установке железа под Linux в этой статье.

ОПРЕДЕЛЯЕМ УСТРОЙСТВА

Как все привыкли устанавливать устройства в Windows? Устанавливают устройство, включают питание, ждут, пока загрузятся системы, и устанавливают драйвер. Лишь для самых древних устройств в базе данных Windows будет драйвер. Важно то, что операционная система сама определяет устройство и используемые им ресурсы. Конечно, все сказанное относится к PnP-устройствам, но когда ты в последний раз видел не PnP-устройство? В Linux для автоматического определения устройств используются специальные утилиты - kudzu или harddrake. В некоторых дистрибутивах используется kudzu или harddrake (преимущественно в дистрибутивах, основанных на Linux Mandrake, и в самом Mandrake, в новых дистрибутивах - harddrake2). Утилита автоматического определения устройств автоматически запускается при загрузке системы. Я рекомендую сразу после установки системы (когда все устройства уже определены и настроены) отключить kudzu (или harddrake, harddrake2) - так запуск системы будет быстрее. Ты же не каждый день устанавливаешь новое устройство? Даже если и так, то все равно ты устанавливаешь новое устройство раз-два в день, а перезагружаешься намного чаще. Поэтому все равно будет выигрыш во времени. После физической установки нового устройства в систему запусти kudzu вручную от имени пользователя root:

```
$ su
# kudzu
```

При запуске harddrake от имени простого пользователя он попросит пароль пользователя root. Введи его и пользуйся harddrake - команду su вводить необязательно.

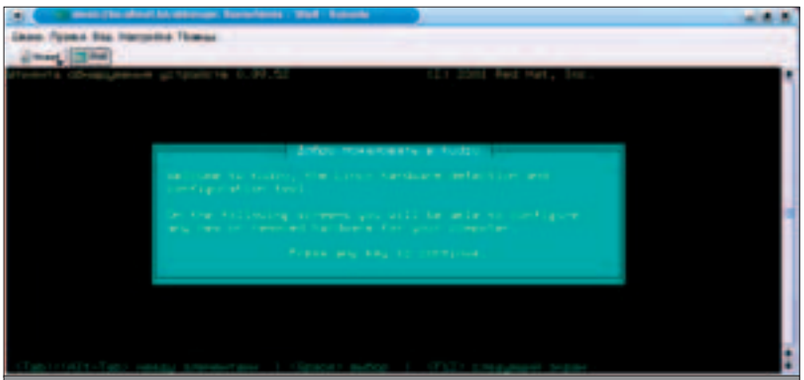
Перед тем как устанавливать новое оборудование, убедись, что ядро поддерживает твое новое устройство. Если нет, пересобирай ядро и включай поддержку нового устройства. Можно со стопроцентной уверенностью сказать, что твое ядро будет поддерживать сетевую плату RTL8139 или любую другую, совместимую с NE2K PCI. А вот о поддержке USB-модема или принтера никаких прогнозов дать нельзя: нужно только запускать программу menuconfig, с помощью которой настраивается ядро или выясняется, какие устройства твое ядро поддерживает, а какие нет. О компилировании ядра читай документы в Сети (HOWTO, FAQ, статьи) или специальные книжки. В принципе, современное ядро 2.6 поддерживает очень много устройств и проблемы могут возникнуть только со следующими типами устройств:

❶. win-модемы, то есть те модемы, которые работают под управлением ОС Windows (я не говорю, что в Linux они вообще не работают, но настраивать его придется долго, а удовольствия от результата вообще получить не удастся);

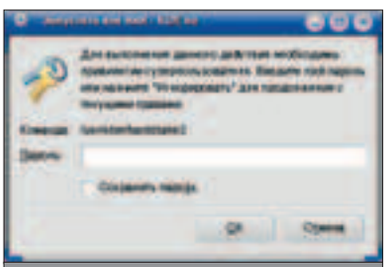
❷. win-принтеры (комментарии те же, что и для win-модемов);

рис. Константин Комардин

В Linux для автоматического определения устройств используются специальные утилиты.



Утилита kudzu

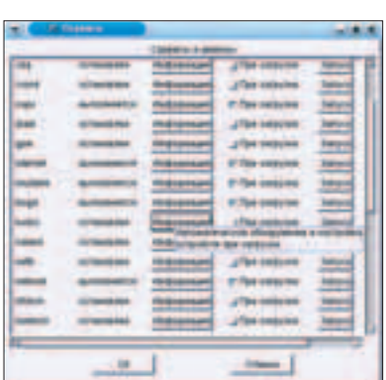


Ввод пароля

нужны для нормальной работы с системой? Прежде всего, видеоадаптер, который настраивается при самой установке системы и который редко просит отдельных настроек. Следующее устройство - это принтер. Он очень легко настраивается конфигуратором printerdrake в Linux Mandrake или redhat-config-printer в Red Hat. Современная база драйверов CUPS (Common Unix Print System) поддерживает большинство принтеров. Практически всю настройку можно произвести с помощью специального для каждого устройства конфигуратора. О таких программах чуть позже, а пока более подробно поговорим о kudzu. Повторюсь: эта утилита запускается при включении компьютера, чем злостно отнимает у тебя время. Рекомендую изгнать ее из автозапуска, а вместо этого запускать ее вручную, когда это понадобится. Выполни команду:

```
# /usr/sbin/drakxservices (или redhat-config-services в Red hat)
```

и отключи автоматический запуск утилиты kudzu.



drakxservices

После установки нового оборудования введи команду:

```
# /usr/sbin/kudzu
```

И тут запустится утилита kudzu, которая сообщит о найденном оборудовании.

Ты можешь согласиться с установкой нового устройства, а можешь отказаться от нее. Задача kudzu в том, чтобы определить, какое устройство установлено, и добавить модули ядра для работы этого устройства. Если ты знаешь точное название модуля, то их можно добавить и вручную с помощью команды insmod (для удаления модуля используется команда rmmod). Kudzu также прописывает модули в файле /etc/modules.conf (чтобы они загружались при запуске системы):

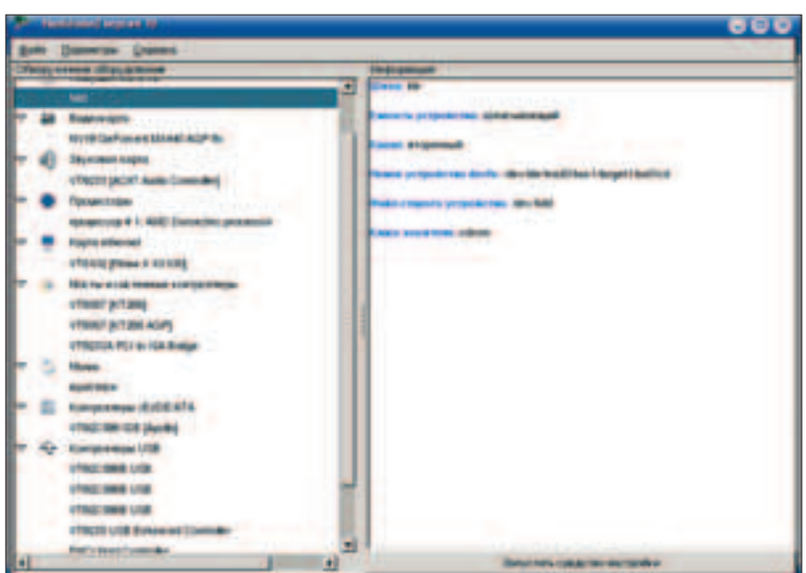
```
pre-install pcmcia_core CARDMGR_OPTS=-f
/etc/rc.d/init.d/pcmcia start
above snd-pcm-oss snd-mixer-oss
alias sound-slot-0 snd-via686
above snd-via686 snd-pcm-oss
alias tap0 ethertap
options tap0 -o tap0 unit=0
```

В этом файле указываются автоматически загружаемые модули и их параметры. Откомпилированные модули хранятся в каталоге /lib/modules. Как добавить устройство вручную, ты узнаешь в следующем пункте - там мы будем вручную добавлять модуль для сетевой платы. Программа kudzu сразу же добавит модули для твоего устройства, поэтому тебе не потребуется по заветам Microsoft перезагружать машину. Подводим небольшие итоги. Алгоритм установки нового оборудования:

1. убедиться, что ядро поддерживает новое устройство (в случае необходимости пересобрать ядро);
2. запустить утилиту /usr/sbin/kudzu (или вручную отредактировать файл /etc/modules.conf (или conf.modules), чтобы установить дополнительные параметры модуля);
3. настроить новое оборудование с помощью соответствующего конфигуратора, например, для настройки сетевой платы использовать netconf.

Подробнее о первом пункте этого алгоритма. Можно запустить menuconfig и посмотреть, какие устройства поддерживает ядро, но при этом пожертвовать немалым количеством времени. Проще будет пойти на сайт Linux Mandrake и посмотреть, есть ли интересующее тебя в базе данных устройств, поддерживаемых Linux. Ничего страшного, если у тебя другой

С помощью пакета modutils можно добавить нужный модуль в ядро во время работы системы.



Конфигуратор Hard Drake

»

времени. Проще будет пойти на сайт Linux Mandrake и посмотреть, есть ли интересующее тебя в базе данных устройств, поддерживаемых Linux. Ничего страшного, если у тебя другой дистрибутив, например, Red Hat - основные устройства те же. Эта база данных доступна по адресу www.mandrakelinux.com/en/hardware.php3. Все, что было сказано про kudzu, распространяется и на harddrake (harddrake2). Его точно так же можно отключать и запускать автоматически.

ПОМОЩНИКИ

Настройка устройства выполняется с помощью соответствующего конфигуратора. Например, printerdrake настраивает принтеры в Mandrake, redhat-config-printer творит то же самое с принтерами в Red Hat. Если ты забыл название конфигуратора, найти нужный конфигуратор очень просто: запусти терминал, введи redhat-config- (если у тебя Red Hat) или drak (если Mandrake) и нажми <Tab>: перед тобой предстанут почти все конфигураторы, доступные в твоей системе. Почему "почти"? Потому что имена некоторых конфигураторов не начинаются с redhat-config или с drak, например, harddrake.

Основные конфигураторы перечислены в таблице.

МОДУЛЬНЫЕ УТИЛИТЫ

Как правило, в ядро включают только самый необходимый код - загрузочную часть, драйвера самых распространенных устройств и дополнительные пакеты. Поддержку остальных устройств обеспечивают модули, которые подгружаются или динамически, или при старте системы. В принципе, можно вкомпилировать в ядро все необходимые драйверы устройств и тем самым получить систему, не использующую модули, но такие системы - уже совсем другая история. С помощью пакета modutils можно добавить нужный модуль в ядро во время работы системы. При этом перезагружать систему не нужно - устройство начнет работать сразу же после загрузки модуля. В первых версиях ядра Linux механизм работы с модулями не был предусмотрен, и ядра тех незапамятных времен содержали в себе код драйверов для ВСЕХ поддерживаемых устройств. Такое решение рациональным не назовешь. Нельзя предусмотреть, какие устройства будут установлены у конечного пользователя, даже если включить в состав ядра драйверы всех возможных устройств. Предположим, что у пользователя X установлена звуковая плата Yamaha, а ядро "знает" еще с десяток звуковых плат кроме этой. Один код будет работать всегда, а остальные десять драйверов будут просто занимать оперативную память. Кстати об оперативной памяти: ты можешь представить себе размер ядра, которое содержит драйверы всех устройств? Вот поэтому разработчики ядра Linux и изобрели механизм динамически загружаемых модулей. Модули хранятся на диске в виде объектных файлов (*.o). При необходимости ядро загружает необходимый ему модуль. Откуда ядро знает, какой модуль нужно загружать, а какой - нет? Списки модулей и передаваемых им параметров хранятся в файле /etc/modules.conf (или /etc/conf.modules - в зависимости от дистрибутива и версии ядра). Вот пример этого файла:

```
alias parport_lowlevel parport_pc
alias usb-controller usb-uhci
alias sound-slot-0 i810_audio
post-install sound-slot-0 /bin/aumix-minimal -f
/etc/.aumixrc -L >/dev/null 2>&1
pre-remove sound-slot-0 /bin/aumix-minimal -f
/etc/.aumixrc -S >/dev/null 2>&1
```

Подробнее о формате файла /etc/modules.conf можно прочитать в справочной системе (man modules.conf). При загрузке система читает этот файл и загружает указанные в нем модули. В нашем случае загружается только модуль i810_audio, поскольку команда alias не загружает модуль, а только устанавливает для него псевдоним. Загрузка модулей из файла modules.conf обеспечивается программой modprobe, которая вызывается из сценария инициализации системы /etc/rc.d/rc.sysinit.

Во время работы системы ты и сам можешь загрузить нужный модуль, для чего существует программа insmod, входящая в состав пакета modutils и очень простая в использовании:

```
# insmod <имя_файла_модуля>
```

Использовать программу может только пользователь root. Просмотреть список загруженных модулей можно с помощью команды lsmod. Вот вывод этой программы:

```
Module Size Used by Not tainted
autofs 12164 0 (autoclean) (unused)
nls_koi8-r 4576 2 (autoclean)
nls_cp866 4576 2 (autoclean)
vfat 12092 2 (autoclean)
fat 37400 0 (autoclean) [vfat]
usb-uhci 24484 0 (unused)
usbcore 73152 1 [usb-uhci]
```

Некоторые модули загружаются не из файла /etc/modules.conf. Например, модули файловых систем загружаются по мере необходимости - при монтировании определенной файловой системы загружается нужный модуль, если, конечно, он есть. Модули nls_koi8-r и nls_cp866 загружаются также при монтировании файловой системы, если указаны опции монтирования iocharset=koi8-r,codepage=866.

Выгрузить модуль предельно просто:

```
# rmmod имя_модуля
```

Программа modinfo позволяет просмотреть более подробную информацию о модуле:

```
# modinfo usbcore

filename: /lib/modules/2.4.18-3/kernel/drivers/usb/usb-
core.o
description: <none>
author: <none>
license: "GPL"
```

Программы insmod, rmmod, lsmod и modinfo входят в состав modutils. Для использования любой из этих программ необходимы права пользователя root.

Ну, вот, собственно, и все. Теперь вперед к грамотной настройке оборудования твоего любимого пингвина!

	Mandrake	Red Hat
Основной конфигуратор	drakconf	setup
Настройка железа	harddrake2	kudzu
Настройка X Window (в том числе монитора и видеокарты)	XFdrake	redhat-config-xfree86
Настройка сети	draknet	redhat-config-network
Настройка клавиатуры	keyboarddrake	redhat-config-keyboard
Настройка мыши	mousedrake	redhat-config-mouse
Настройка принтера	printerdrake	redhat-config-printer

Основные конфигураторы Mandrake и Red Hat

Модули хранятся на диске в виде объектных файлов.

МЫ ЗНАЕМ О ЛУЧШИХ ИГРАХ ВСЕ!
...И ДАЖЕ ЧУТЬ БОЛЬШЕ

В ДЕКАБРЬСКОМ НОМЕРЕ:

SILENT HILL 4: THE ROOM
- прохождение игры
- описание оружия
- описание всех концовок

MYST IV: REVELATION
- полное прохождение
- рассказ о персонажах

ROME: TOTAL WAR
- общие советы по игре
- описание юнитов

WARHAMMER 40,000: DAWN OF WAR
- полное прохождение
- описание юнитов

CD: Видеоуроки по прохождению и русскоязычная база кодов и прохождений

«ПУТЕВОДИТЕЛЬ: PC ИГРЫ»
ЖУРНАЛ КОДОВ И ПРОХОЖДЕНИЙ ДЛЯ ЛУЧШИХ КОМПЬЮТЕРНЫХ ИГР

Андрей Семенюченко (semu@rbcmail.ru)

ВЫЛЕЗАЕМ В ИНТЕРНЕТ

НАСТРОЙКА СЕТИ В LINUX

Формирование Сети включает в себя много этапов от проектирования до проверки ее работоспособности и детальной настройки сервисов. Ты думаешь, что это долго и трудно? Зря! Настройка сетевых соединений в Linux - очень увлекательное и познавательное занятие.

ФОРМИРОВАНИЕ СТРУКТУРЫ СЕТИ

Перед физическим подключением кабелей к узлам сети и до настройки сетевых сервисов прежде всего необходимо тщательно формализовать будущую как логическую, так и физическую структуру Сети. Так что сначала ты должен определится с теми аппаратными средствами и сетевыми технологиями, которые собираешься использовать при создании своей сетки. В зависимости от того, собираешься ли ты соединить два компьютера между собой и вывести их в интернет или же собрать региональную локальную сеть, используемое коммуникационное оборудование будет в некоторой степени отличаться. При проектировании необходимо учитывать такие факторы, как пропускная способность сети, резервное оборудование, дополнительные обходные маршруты, нагрузка на коммуникационное оборудование и др.

НАСТРОЙКА УЗЛОВ СЕТИ

Раз уж ты добрался до настройки непосредственно рядовых компьютеров в сети, значит, процесс проектирования сети и физического подключения уже позади. Процесс же настройки ethernet-сети включает два этапа:

1. настройка сетевого интерфейса;
2. настройка сетевых параметров.

Сегодняшние Linux-дистрибутивы поддерживают большинство современных сетевых плат с подключением через ISA, PCI, PCMCIA и USB-интерфейсы. Все адаптеры, за исключением адаптеров для ISA-шины, не требуют специальной настройки и определяются дистрибутивом автоматически. Затруднения могут возникнуть при попытке добавить сетевую карту уже после того, как система установлена. В этом случае нужно установить модуль, подходящий для сетевой платы.

Возможно, драйвер для сетевого адаптера уже имеется в твоем дистрибутиве. Модули к сетевухам лежат в директории /lib/modules/версия_ядра/kernel/drivers/net/.

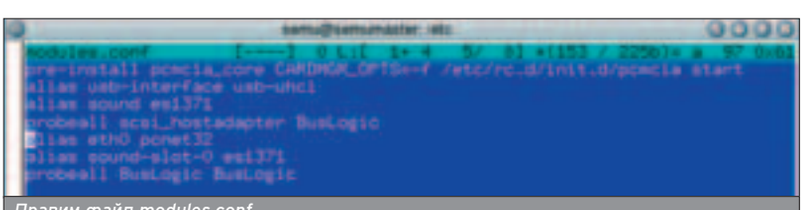
Если драйвера нет, тогда иди на сайт разработчика карты и скачай нужный модуль для используемой версии ядра. Потом нужно будет прописать название модуля в файле настроек /etc/modules.conf. Например, для адаптера PCI Fast Ethernet на основе Realtek RTL8139(A) в modules.conf прописываем строку:

```
alias eth0 pcnet32
```

Эта запись означает, что устройству, именуемому eth0, соответствует модуль pcnet32.

Всегда полезно знать, за что отвечает тот или иной конфигурационный файл, поэтому попытаемся настроить сеть вручную, без новомодных утилит с удобным графическим интерфейсом (Draknet, Network Administration Tool). Первым делом лезем в /etc/sysconfig/network-scripts и создаем там файл ifcfg-eth0. В него с помощью твоего любимого текстового редактора нужно запихнуть примерно следующий текст:

```
DEVICE=eth0
ONBOOT=yes
IPADDR=192.168.10.20
NETMASK=255.255.255.0
NETWORK=192.168.10.0
BROADCAST=192.168.10.255
```



Правим файл modules.conf

Фактически, этот файл дает указание системе поднять во время ее загрузки устройство eth0 с IP-адресом 192.168.10.20, сетевой маской 255.255.255.0, сетевым адресом 192.168.10.0 и широковещательным адресом 192.168.10.255.

Теперь идем в каталог /etc/sysconfig/ и создаем файл network со следующим содержанием:

```
NETWORKING=yes
HOSTNAME=yourhostname
DOMAINNAME=yourdomain.domain
GATEWAY=192.168.10.1
```

Как ты понимаешь, в этом файле прописывается dns-имя твоего хоста, название домена и IP-адрес шлюза.

Последним шагом является заполнение DNS-серверов в файле /etc/resolv.conf.

```
search yourdomain.domain
nameserver 208.185.249.250
nameserver 192.168.10.152
```

Несколько па на клавиатуре, и локальная сеть настроена!

ПРОКЛАДЫВАЕМ ДОРОГУ В ИНТЕРНЕТ

По большому счету, на сегодняшний день ты можешь подключиться к интернету тремя способами:

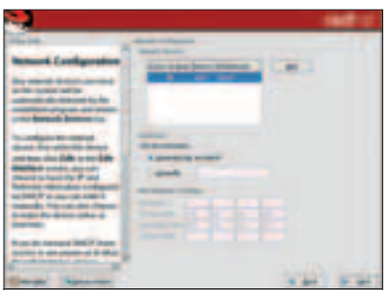
1. через медленный, ненавистный многим, но дешевый Dial-up;
2. с помощью xDSL-оборудования;
3. по выделенной линии через локальную сеть.

Конечно, существуют и другие способы подключения, но они мало распространены из-за сложности и дороговизны и требуют отдельного обсуждения.

ШАГАЕМ ПО ВЫДЕЛЕННОЙ ЛИНИИ

Начнем с последнего, потому как для этого способа уже все подготовлено. Сетевая карта уже работает, а другого оборудования устанавливать тут не понадобится.

Настройка сети с помощью утилиты draknet из состава Mandrake



Средство настройки сети от RedHat - Network Administration Tool

Настройка выхода в интернет в локальной сети может быть реализована несколькими способами, в зависимости от того, как организована сетка. Способ первый, самый распространенный: нужно всего лишь прописать IP-адрес шлюза (что уже сделано), куда пакеты будут идти по умолчанию, если они не адресованы в рамках твоей локальной сети. В нашем случае - 192.168.10.0. При этом во внешней сети на одном из пограничных серверов настроен NAT (Network Address Translation), маскарадинг или нечто подобное. Что именно и как - нас не волнует, главное, что до интернета добрались.

Способ второй: использовать для выхода в интернет прокси-сервер. Тогда нужно будет или внести адрес прокси в настройки каждого приложения, использующего соединение с глобальной Сетью, или (что проще) создать и заполнить переменные окружения http_proxy и ftp_proxy, а затем экспортировать их командой export:

```
http_proxy=http://my_login:my_password@my_proxy.mydomain.domain:poxy_port
ftp_proxy=ftp://my_login:my_password@my_proxy.mydomain.domain:poxy_port
```

СТАРЫЙ ДОБРЫЙ ДИАЛ-АП

Значит, ты все-таки купил модем, чтобы доставать своих домашних постоянно занятой телефонной линией. Но это твое решение, его уважаю я, и, видимо, его придется уважать твоей семье. Думаю, ISP (Internet Service Provider) ты выберешь сам: их навалом у нас в стране, осталось только определиться с тарифом и настроить Linux.

Вот то, что понадобится (часть этой информации размещена на карточке провайдера):

- номер телефона провайдера;
- тип набора номера (импульсный или тоновый);
- login (имя пользователя);
- password (пароль пользователя);
- IP-адрес первичного сервера DNS;
- IP-адрес вторичного сервера DNS;
- возможно, тип аутентификации (скорее всего PAP, но может быть и CHAP);
- возможно, имя домена (ISP_domain.ru).

Надеюсь, у тебя уже есть пакет ppp, если нет, то быстро качай и ставь.

Настроить соединение можно опять же как с помощью консольных и графических утилит, автоматизирующих работу пользователя, так и путем ввода информации в соответствующие файлы вручную.

Все конфигурационные файлы для настройки ppp-соединения находятся в директории /etc/ppp. Поскольку, скорее всего, там есть примеры настроек, останется добавить только самое необходимое.

В файле /etc/ppp/scripts/ppp-on:

```
# Номер модемного пула провайдера
TELEPHONE=1111111
ACCOUNT=login          # Регистрационное имя
PASSWORD=password      # Пароль
LOCAL_IP=0.0.0.0       # Назначается динамически
# IP-адрес с внешней стороны. Обычно: 0.0.0.0
REMOTE_IP=0.0.0.0
NETMASK=255.255.255.0  # Маска подсети
```

В файле /etc/ppp/options нужно указать домен провайдера:

```
lock
domain ISP_domain.ru
```

После настройки соединения подключаться к интернету можно разными способами, например, прибегнув к услугами утилиты kppp с графическим интерфейсом KDE или из командной строки: /sbin/ifup ppp0.

ТЕХНОЛОГИЯ DSL

DSL - набор различных технологий, позволяющих организовать цифровую абонентскую линию. Его преимущества очевидны: высокая скорость, использование уже существующих телефонных линий, неполная занятость телефонной линии (по телефону поболтать можно одновременно), дешевизна. Наиболее распространена технология ассиметричной цифровой абонентской линии ADSL.

Для настройки сетевого подключения по adsl-модему нужен пакет rp-pppoe от Roaring Penguin Software Inc.

Поднять ADSL чуть ли не проще, чем в Windows: нужно только запустить adsl-setup и вбить следующее:

USERNAME - логин для соединения;
PASSWORD - пароль для соединения;
INTERFACE - интерфейс модема (у популярного adsl-модема Zyxel OMNI USB - это nas0);
DNS - адрес dns-сервера (можно оставить пустым).

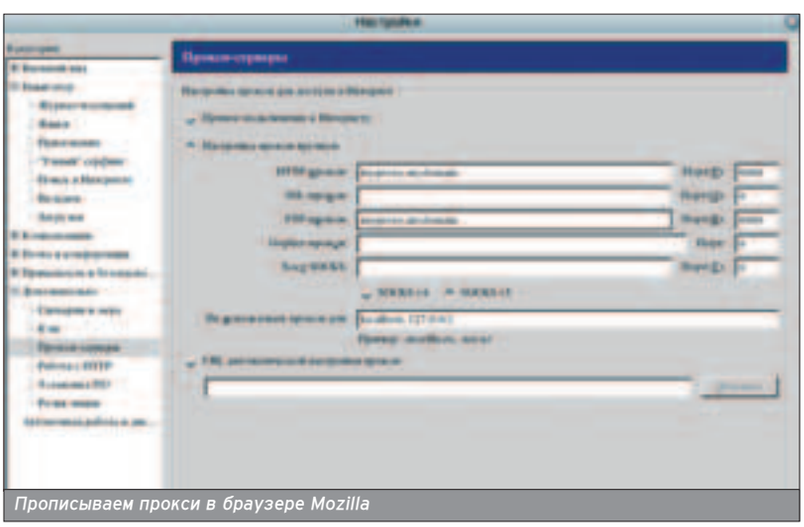
Теперь ты можешь воспользоваться скриптами adsl-start, adsl-stop и adsl-connect для установки соединения.

ПРОВЕРКА РАБОТОСПОСОБНОСТИ СЕТИ

В Linux для того, чтобы убедиться в работоспособности сети, существует множество диагностических утилит, таких как ping, nslookup, traceroute, tracepath, ifconfig, route и др.

Обычно проблемы с сетью сначала лучше решать в компании с утилитой ping. Если пакеты проходят, возможно, проблемы с dns-сервером, тогда в качестве IP-адреса можно указать адрес dns-сервера или попытаться разрешить dns-имя с помощью утилиты nslookup. Иногда кажется, что все настроено, а интернет все равно отказывается работать. В этом случае попробуй прописать шлюз по умолчанию. Например, для устройства ppp0 сделай следующее: route add default ppp0.

Надеюсь, этих сведений о настройке сети тебе хватит, чтобы наконец вылезти из своего любимого пингвина в интернет. Будут проблемы - пиши, постараюсь помочь.



Прописываем прокси в браузере Mozilla

Если в системе установлены две одинаковые сетевые карты, для их настройки достаточно загрузить один драйвер - он будет обслуживать оба устройства.

При наличии нескольких сетевых адаптеров устройства в Linux именуются по порядку загрузки драйверов, то есть первый - eth0, второй - eth1, третий - eth2 и т.д.

Запустить утилиту Network Administration Tool можно из командной строки командой redhat-config-network.

Антон Карпов, toxa@cterra.ru

ГРАНИЦА НА ЗАМКЕ

ПОДНИМАЕМ БЕЗОПАСНЫЙ И ФУНКЦИОНАЛЬНЫЙ ШЛЮЗ ДЛЯ ЛОКАЛЬНОЙ СЕТИ

В Москве вовсю свирепствует "Стрим" - ночной кошмар домашних локальных сетей. Однако в других городах нашей необъятной Родины картина не такая радужная: даже в моем родном Питере полно районов, куда не ступала нога "домашнего" провайдера. А значит, самопальные локальные сети все еще живут и здравствуют. Как правило, строители таких сетей считают, что грамотно проложенные кабели и свитчи - это единственная основа надежной работы сети, а все остальное можно переложить на плечи Wingate. Разумеется, это мнение неверно, и ты еще можешь спасти свою сеть, если отделишь ее от интернета надежным шлюзом.

СУРОВЫЙ БЫТ ДОМАШНИХ ЛОКАЛЬНЫХ СЕТЕЙ

Основное отличие домашней локальной сети от маленькой корпоративной сети в особенностях поведения пользователей. Пользователи корпоративной сети обязаны соглашаться со всем, что прописано в документе под названием "Корпоративная политика", и терпеть ограничения на доступ к внешним почтовым серверам, авторизацию на прокси-сервере и т.д. В "дикой" локальной сети людям нужен интернет без лишних заморочек, к тому же, как правило, здесь обязательно обитают доморощенные "хакеры", которых хлебом не корми дай украсть чужого трафика. Кроме того, народ, который платит за скачанные мегабайты, вполне законно хочет регулярно отслеживать свою статистику. Наконец, пользователи вряд ли согласятся платить за дорогие свитчи с фильтрацией по портам, VLAN'ами и за мощное железо. Разумеется, провайдер, который протянет "последнюю милю", вряд ли будет всем этим озадачивать себя. Итак, на тебя смотрит видавший виды компьютер, готовящийся стать неприступным шлюзом. Первым делом водружаем на него OpenBSD и обновляем до актуальной -stable версии (на момент написания статьи таковой была 3.6-stable) предварительно распаковав в /usr/src исходники ядра и системы (пакеты src.tar.gz и sys.tar.gz с ftp://ftp.openbsd.org/pub/OpenBSD/3.6):

```
# setenv CVSROOT
anoncvs@anoncvs.ca.openbsd.org:/cvs
# cd /usr/src
# cvs -qz9 up -rOPENBSD_3_6 -Pd
```

Если в ядре начиная еще времен его релиза обнаруживали критические уязвимости, сначала пересобери его. В противном случае делать это настоятельно не рекомендуется, да и при пересборке крайне нежелательно менять конфигурацию ядра оставив дистрибутивный GENERIC:

```
# cd /usr/src/sys/arch/i386/conf
# config GENERIC
# cd ../compile/GENERIC
# make depend %26%26 make
# mv /bsd /bsd.old %26%26 cp bsd /bsd
# reboot
```

Теперь пересоберем userland:

```
# cd /usr/src
# make -k cleandir %26%26 make obj %26%26 make build
# reboot
```

Если машина слишком слабая, можно ограничиться наложением патчей с www.openbsd.org/errata.html. Это, в общем, и есть рекомендованный способ обновления OpenBSD. А кто нам мешает поразвлечься с пересборкой системы? ;)

```
RULES="$RULES\npass out inet6 proto icmp6 all
icmp6-type routersol"
RULES="$RULES\npass in inet6 proto icmp6 all icmp6-
type routeradv"
```

Очевидно, pf ругается на отсутствие поддержки inet6 и падает лапками кверху. Закомментируй эти строчки или просто откажись от сборки своего ядра - на это вряд ли есть веские причины.

Сконфигурировать OpenBSD не сложнее, чем установить его. Классическое решение - шлюз, отделяющий локальную сеть с внутренней адресацией (сети 192.168.0.0/16, 172.16.0.0/12, 10.0.0.0/8 согласно RFC1918) от интернета. В этом случае нужно сконфигурировать оба интерфейса, включить маршрутизацию между интерфейсами (форвардинг) и настроить Network

Сайт OpenBSD. Но OpenBSD - это не только безопасная ОС...

OpenCVS - проект разработчиков OpenBSD

PFSysinfo - еще одно средство мониторинга pf и системы, с графиками, отрисованными pflog'ом

Address Translation (NAT). В OpenBSD сетевые карты настраиваются путем занесения в файл /etc/host-name.%26lt;имя_интерфейса%26gt; необходимой информации.

Внешний адрес:

```
# cat /etc/hostname.vr0
inet 62.89.2XX.XX 255.255.255.192 NONE
```

Внутренний адрес:

```
# cat /etc/hostname.vr1
inet 192.168.0.1 255.255.255.0 NONE
```

В /etc/sysctl.conf раскомментируй строчку:

```
net.inet.ip.forwarding=1
```

Информация о шлюзе провайдера (default gateway) заносится в файл /etc/mygate:

```
# cat /etc/mygate
62.89.2XX.1
```

Для пользователей локальной сети шлюзом по умолчанию станет внутренний интерфейс машины - 192.168.0.1. Наконец, если провайдер внес в свою зону имен A-запись для твоего шлюза (или если ты собираешься поднять свой dns-сервер, что вряд ли потребуется для домашней сети), то внеси имя машины в файл /etc/myname:

```
# cat /etc/myname
puffy.toxahost.ru
```

Если имя не обслуживается ни одним dns-сервером, то внеси то же имя в /etc/hosts:

```
62.89.2XX.XX puffy.toxahost.ru
```

Некоторые провайдеры предоставляют клиентам только внешние IP-адреса. В таком случае на твою локальную сеть будет выделена подсеть из диапазона адресов, принадлежащих провайдеру, а шлюз будет выполнять роль моста (bridge), прозрачного для сети. Поднять мост на OpenBSD просто как пять копеек:

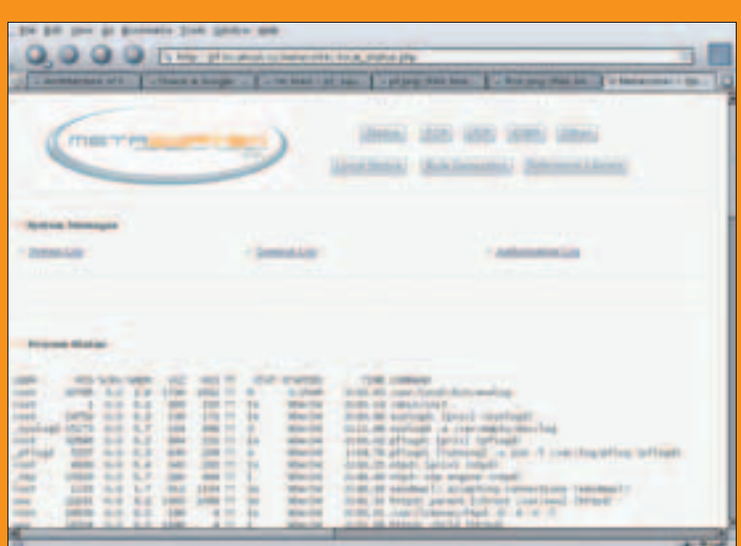
```
# cat /etc/bridgename.bridge0
add vr0
add vr1
blocknonip vr0
blocknonip vr1
up
```

rge vr0 и vr1 - интерфейсы, которые мы будем "бриджить". Хоть мост работает на втором уровне модели TCP и имеет дело исключительно с MAC-адресами, ему, тем не менее, можно присвоить IP-адрес для удаленного доступа. Строго говоря, "прозрачный мост" (transparent bridge) и мост с присвоенными адресами - разные вещи, так как ведут себя по-разному в некоторых случаях. В нашем случае требуется удаленный доступ к машине, но нет нужны присваивать адреса обоим интерфейсам, поэтому /etc/hostname.vr0 оставим как есть, а в /etc/hostname.vr1 пропишем просто поднятие интерфейса:

```
# echo up %26gt; /etc/hostname.vr1
```

Шлюзом по умолчанию для клиентов в таком случае будет роутер провайдера.

■ Если ты чувствуешь себя настоящим мачо и вопреки всем предупреждениям решился пересобрать ядро со своим конфигурационным файлом, будь внимателен. Во-первых, четко следуй инструкциям **www.openbsd.org/faq/faq5.html**. Во-вторых, будь готов ко всему ;). Так, например, реализация непременно возникающего желания выкинуть поддержку IPv6 приведет к неработоспособности pf. А все из-за этих строчек в /etc/rc, осуществляющих первоначальное конфигурирование пакетного фильтра:

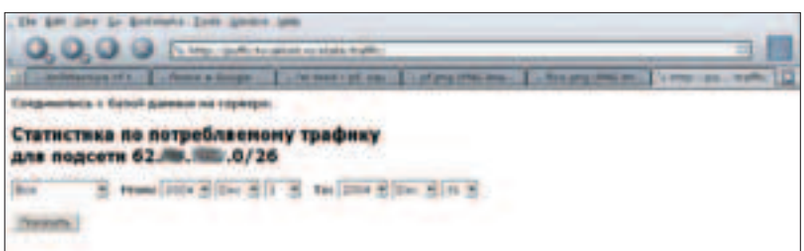


Metacortex - удобное средство удаленного мониторинга pf и системы

ФУНКЦИОНАЛЬНОСТЬ ПРЕЖДЕ ВСЕГО

■ Определимся с тем, что будет крутиться на шлюзе, кроме стандартного sshd для удаленного администрирования. Для получения почты с отчетами о работе системы мы запустим pop3-сервер pop'a3d (входит в штатную поставку), для подсчета трафика соорудим систему trafd/mysql. Экономить на трафике путем кеширования web-страниц будем с помощью прокси-сервера squid. Наконец, чтобы обеспечить контроль над пользователями, в качестве метода "на коленке" »

OpenBGPD - проект разработчиков OpenBSD



Заглавная страница статистики

для защиты сети от любителей менять IP-адрес на адрес соседа поставим arpwatch, который будет вести таблицу записей "IP-MAC" и сигнализировать об аномалиях. Разумеется, для "хакера", охочего до чужого трафика, не составит труда поменять и MAC-адрес. Но по большому счету, более-менее адекватную защиту может обеспечить разве что фильтрация MAC на порту свитча (а такие приборы стоят денег) или организация VPN. Но хочешь ли ты ходить из квартиры в квартиру и показывать недалеким пользователям, где у них в Windows настраиваются политики IPSec?

В нашей дикой локальной сети можно лишь сделать попытку борьбы с вопиющим беспорядком. Заключительным аккордом станет поднятие web-сервера для обеспечения пользователям web-доступа к статистике по трафику и пакетного фильтра для спокойной работы сети. В качестве pop3-сервера будем использовать безопасный и надежный popa3d (являющийся частью OpenBSD). Так как все системные отчеты и вывод cron'а складываются в почту root'у, а мы работаем из-под непривилегированного пользователя, внесем изменения в /etc/mail/aliases и пересоберем /etc/mail/aliases.db:

```
# echo root: toxa %26gt;%26gt; /etc/mail/aliases
%26%26 newaliases
```

Теперь вся root'овая почта будет сыпаться пользователю toxa. Допиши в конец файла /etc/rc.local запуск popa3d в standalone-режиме:

```
if [ -x /usr/sbin/popa3d ]; then
echo -n ' popa3d'; /usr/sbin/popa3d -4D
fi
```

Теперь ставим arpwatch. Все необходимые программы будем ставить из прекомпилированных пакетов, так что если у тебя нет диска с содержимым ftp.openbsd.org/pub/OpenBSD/3.6/packages/i386/, то пропиши переменную окружения PKG_PATH соответствующим образом:

```
# export
PKG_PATH=ftp://ftp.openbsd.org/pub/OpenBSD/3.6/pack-
ages/i386/
```

Затем добавляй нужные пакеты

```
# pkg_add arpwatch-2.1a13.tgz
```

Запуск arpwatch на внутреннем интерфейсе также пропишем в /etc/rc.conf.local:

```
if [ -x /usr/local/sbin/arpwatch ]; then
echo -n ' arpwatch'; /usr/local/sbin/arpwatch -i vr0
fi
```

Arpwatch после запуска начнет коллекционировать информацию в файл /var/arpwatch/arp.dat. Поначалу твой почтовый ящик будет заполнен сообщениями о том, что в сети появились новая станция:

```
From: Arpwatch
%26lt;arpwatch@puffy.toxahost.ru%26gt;
To: root@puffy.toxahost.ru
Subject: new station

hostname: %26lt;unknown%26gt;
ip address: 62.89.2XX.XX
ethernet address: 0:0:39:84:21:e3
ethernet vendor: TOSHIBA CORPORATION
timestamp: Thursday, November 18, 2004 13:49:39
+0300
```

Если она определит несоответствие ip-mac уже существующей записи в таблице, будет выслано предупреждение "changed ethernet address". Конечно, предупреждение не заблокирует сессию хацкера, однако по факту легко будет установить хулигана. Будем считать, что нам этого достаточно. Теперь ставим прокси-сервер. К сожалению, та версия squid, которая находится в портах, уже попахивает тухлятиной, и нам придется собирать

OpenNTPD - проект разработчиков OpenBSD

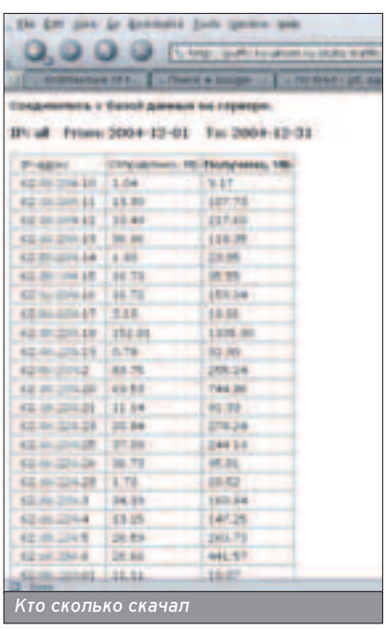
МНЕНИЕ ЭКСПЕРТА

■ Андрей Матвеев, редактор рубрики "Юниксоид" журнала "Хакер" (andrushock@real.xakep.ru)

Трансляция сетевых адресов, перенаправление пакетов между сетевыми интерфейсами, фильтрация входящих и исходящих запросов... Можно еще очень долго перечислять функции, которые возложены на систему, обеспечивающую взаимодействие домашней/корпоративной локальной сети с интернетом. Не стоит также забывать, что сейчас затмевают собой все другие проблемы и становятся все острее вопросы защиты клиентских компьютеров от несанкционированного доступа. При поднятии шлюза важно выбрать операционную систему именно со следующими достоинствами: обладание грамотной реализацией стека TCP/IP и мощным файрволом с гибким синтаксисом правил; наличие в составе ОС подавляющего большинства сетевых служб, запускаемых от имени непривилегированного пользователя и/или в измененном корневом каталоге (chroot), что позволило бы максимально снизить ущерб при возможном взломе.

squid руками. Так как в релизах squid часто находят уязвимости (к счастью, несерьезные), то к последнему релизу, как правило, прилагается ворох патчей. Но мы сделаем проще, а именно скачаем ежедневно генерируемый релиз и поставим:

```
# wget www.squid-cache.org/Versions/v2/2.5/squid-2.5.STABLE7-20041219.tar.gz
# tar xzf squid-2.5.STABLE7-20041219.tar.gz
# cd squid-2.5.STABLE7-20041219
# ./configure --prefix=/usr/local/squid --sysconfdir=/etc/squid --enable-storeio="ufs diskd" --enable-poll --enable-pf-transparent --disable-ident-lookups --enable-removal-policies="lru heap" --disable-wccp --enable-err-language=Russian-koi8-r
# make %26%26 make install
# groupadd -g 515 _squid
# useradd -d /nonexistent -s /sbin/nologin -u 515 -g 515 -c "Squid Account" _squid
# chown _squid:_squid /usr/local/squid/var/cache
# chown _squid:_squid /usr/local/squid/var/logs
# chgrp _squid /dev/pf %26%26 chmod g+rw /dev/pf
```



Кто сколько скачал

Ключевой момент этих шаманств - поддержка pf transparent, прозрачного проксирования трафика, который мы будем заворачивать на squid с помощью pf. Именно для этого мы даем возможность squid'у читать и писать в /dev/pf (последние строчки). В результате пользователи даже не узнают о том, что их трафик экономит наш кеширующий прокси-сервер. После установки открываем /etc/squid/squid.conf и вносим туда следующие изменения:

```
http_port 127.0.0.1:3128
icp_port 0
http_access deny to_localhost
acl our_networks src 192.168.0.0/24
http_access allow our_networks
httpd_accel_host virtual
httpd_accel_port 0
httpd_accel_with_proxy on
httpd_accel_uses_host_header on
cache_effective_user _squid
cache_effective_group _squid
```

Прописываем автозапуск squid в /etc/rc.local:

```
if [ -x /usr/local/squid/bin/squid ]; then
echo -n ' squid'; /usr/local/squid/bin/squid -D
fi
```

Первый раз нужно запустить squid с параметром -z для создания кеша:

```
# /usr/local/squid/sbin/squid -z
```

Теперь пришла очередь кеширующего DNS. Вопреки популярной практике мы не будем настраивать dns-сервер BIND, а воспользуемся более безопасным и шустрым dnscache из пакета djbdns. Подробнее о djbdns и установке его составных частей читай в этом номере, а здесь я отмечу лишь одну особенность: укажем dnscache работать в качестве форвардера, пересылая все запросы к dns-серверу провайдера и кешируя результат:

```
# echo prov.dns.serv.ip %26gt; /etc/dnscache/root/servers/@
# echo 1 %26gt; /etc/dnscache/env/FORWARDONLY
```

Из-за недостатков ftp-протокола у пользователей могут возникнуть проблемы при доступе к ftp-серверам, которые решает запускаемая из inetd ftp-proxy, на которую мы будем заворачивать ftp-соединения с помощью pf:

```
# grep ftp-proxy /etc/inetd.conf
127.0.0.1:8021 stream tcp nowait root /usr/libexec/ftp-proxy ftp-proxy
```

СЧЕТ, ПОЖАЛУЙСТА

■ Пользователи будут в восторге, если ты их порадуешь индивидуальной статистикой потребления трафика. Apache + trafd + mysql могут подарить им такую радость. Ставим trafd и mysql:

```
# pkg_add trafd-3.0.1.tgz
# mkdir -p /var/trafd/traffic_plain
# mkdir -p /var/trafd/traffic

# pkg_add mysql-server-4.0.20.tgz
# /usr/local/bin/mysql_install_db
```

Прописываем их в автозапуск, /etc/rc.local:

```
if [ -x /usr/local/sbin/trafd ]; then
echo -n ' trafd'; /usr/local/sbin/trafd -i vr0
fi

if [ -x /usr/local/bin/mysqld_safe ]; then
echo -n ' mysqld'; /usr/local/bin/mysqld_safe --user=_mysql --datadir=/var/mysql %26
fi
```

Поставим необходимые модули для работы perl с mysql:

```
# pkg_add p5-Net-MySQL-0.08.tgz
```

Подправим конфигурационный файл mysql, чтобы база данных слушала только локальный socket и не принимала соединения из сети, а также изменим местоположение pid-файла, чтобы его мог найти httpd, который будет записан в /var/www:

```
# cp /usr/local/share/mysql/my-medium.cnf /etc/my.cnf
# vi /etc/my.cnf

[client]
port = 3306
socket = /var/www/var/run/mysql/mysql.sock

[mysqld]
port = 3306
socket = /var/www/var/run/mysql/mysql.sock
skip-networking
```

Зададим пароль root'у на доступ к mysql (по умолчанию он пуст), созда- »

дим отдельную базу для статистики и укажем использование непривилегированным пользователем ipacct:

```
# mysqladmin -u root -p password 'securepass'
# mysql -u root -p
Enter password: %26lt;securepass%26gt;
mysql%26gt; create database traffic;
mysql%26gt; use traffic;
mysql%26gt; create table yesterday (src_ip char(16),
src_port int, dst_ip char(16), dst_port int, proto int, bytes
bigint);
mysql%26gt; create table traffic_tmp (ip char(16), sent
bigint default 0, recv bigint default 0);
mysql%26gt; create table traffic (dt date, ip char(16),
sent bigint default 0, recv bigint default 0);
mysql%26gt; grant delete,insert,select,update on traf-
fic.* to 'ipacct'@'localhost' identified by 'ipacctpass-
word';
mysql%26gt; flush privileges;
mysql%26gt; quit
```

Для обработки статистики, накопленной trafd, существуют специальные скрипты. Авторство их, по всей видимости, принадлежит написавшему статью www.tmeter.ru/misc/trafd. Ищи на нашем диске эти скрипты, модифицированные специально под OpenBSD. Распакуй их в /root/scripts/trafd/, замени имя интерфейса vr0 на свое и пропиши их запуск в crontab:

```
# crontab -e
# dump to tempfile in case to recovery from it
*/15 * * * * /usr/local/sbin/trafdump vr0
# dump to binary file to be rotated each day
55 23 * * * /usr/local/sbin/trafsave vr0
# rotate binary files each day
56 23 * * * /root/scripts/trafd/trafrotate.sh
# convert from binary to plain text log file
57 23 * * * /root/scripts/trafd/traflog.sh
# put all this shit into database
58 23 * * * /root/scripts/trafd/db_update.sh
```

Итогом всей этой работы станет ежедневное любезное складирование статистики за весь день в файл /var/trafd/traffic_plain/{дата}.{имя_интерфейса} в текстовом удобочитаемом виде. Этот файл будет обрабатываться скриптом, а информация - записываться в базу данных. Теперь нужно обеспечить пользователей удобным web-интерфейсом. И тут на сцену выходит apache и php-скрипты из вышеупомянутого набора.

```
# pkg_add php4-core-4.3.10.tar.gz
# pkg_add php4-mysql-4.3.10.tar.gz
# /usr/local/sbin/phpxs -s
```

Apache - то немногое, что мне нравится в OpenBSD. И хоть формально это httpd версии 1.3.29, на самом деле Apache в базовой поставке OpenBSD сильно отличается от такового с httpd.apache.org. В нем исправлено множество ошибок, и в целях безопасности он по умолчанию запускается в chroot(). Однако это и создает дополнительные проблемы. Например, после установки php нужно проделать следующее:

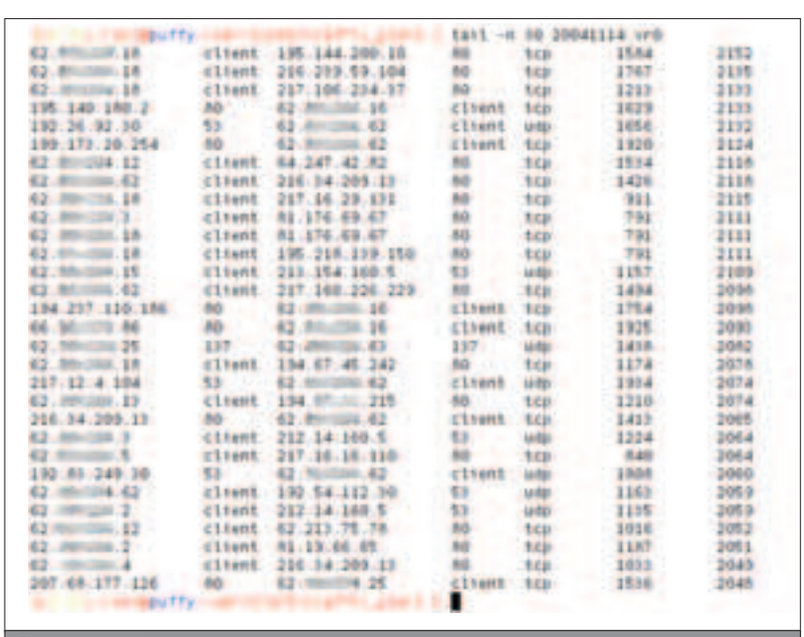
```
# cp /usr/local/share/doc/php4/php.ini-recommended
/var/www/conf/php.ini
```

Затем поправим php.ini:

```
# vi /var/www/conf/php.ini

safe_mode_exec_dir = /var/www/
expose_php = Off
include_path = ".:/pear/lib:/var/www/pear/lib"
extension_dir = "/var/www/lib/php/modules"
safe_mode_gid = Off
allow_url_fopen = Off
```

Поправим /var/www/conf/httpd.conf:

```
LoadModule php4_module
/usr/lib/apache/modules/libphp4.so
DirectoryIndex index.html index.php
AddType application/x-httpd-php .php
```



Статистика trafd

Потом скопируем скрипты index.php oper.php procs.inc в /var/www/htdocs.

Apache - часть системы, по умолчанию он отключен. А чтобы не портить /etc/rc.conf, создавай /etc/rc.conf.local следующего содержания:

```
#!/bin/sh -

ntpd_flags=""
httpd_flags=""
```

Заодно был включен демон точного времени OpenNTPD, который будет синхронизировать системные часы с внешних time-серверов. Знание точного времени никогда не повредит. Запускаем apache:

```
# apachectl start
```

Теперь пользователи могут любоваться статистикой не покидая браузер ;).

СТРОИМ ОГНЕННУЮ СТЕНУ

Наконец, настроим пакетный фильтр pf, который по умолчанию отключен.

```
echo pf=YES %26gt;%26gt; /etc/rc.conf.local
```

Займемся pf и его конфигурационным файлом /etc/pf.conf. Политика будет простая: пропускать все наружу, блокировать все попытки соединения с машинами локальной сети извне. Кроме того, мы откроем доступ к шлюзу отовсюду по ssh, чтобы в случае чего залогиниться на него из любой точки земного шара и разрулить проблемы, а также с определенной машины откроем доступ по pop3, чтобы получать отчеты о работе системы. Не забудем и про заворачивание на squid и ftp-proxy пакетов, идущих на www- и ftp-серверы соответственно. На десерт мы прикрутим ALTQ - систему Quality Of Service, которая упорядочивает пакеты по приоритетам согласно указанным правилам.

```
# vi /etc/pf.conf
# $OpenBSD: pf.conf,v 1.28 2004/04/29 21:03:09
frantzen Exp $
# See pf.conf(5) and /usr/share/pf for syntax and
examples.
# Remember to set net.inet.ip.forwarding=1 and/or
net.inet6.ip6.forwarding=1
# in /etc/sysctl.conf if packets are to be forwarded
between interfaces.
# Rules must be in order:
# options, normalization, queueing, translation, filter-
ing

# 1. MACROSES AND TABLES (options)
# Описываем наши интерфейсы, подсеть, и свою
машину в локалке
ext_if="vr0"
int_if="vr1"
loop_if="lo0"
adminbox="192.168.0.10"
subnet="192.168.0.0/24"
```

■ Учти, что фильтрация пакетов на мосту отличается от таковой на обычном шлюзе. Так, в случае бриджа не будут работать rdr и nat правила, за исключением такой ситуации, в которой каждому интерфейсу моста присвоен IP-адрес. Так что transparent bridge и transparent proxy несовместимы. Кроме того, так как мост соединяет одну и ту же сеть, есть смысл фильтровать пакеты только на одном (внешнем по отношению к локальной сети) интерфейсе, пропуская все на внутреннем.

OpenSSH - проект разработчиков OpenBSD

```
  # Порты IM-служб (icq, jabber, aol)
  im_ports = "{ 1863, 5190, 5222, 6667 }"
  # Не маршрутизируемые в интернет адреса
  table %26lt;priv_nets%26gt; { 127/8, 192.168/16,
172.16/12, 10/8 }

  # 2. PF SETTINGS (normalisation)
  # Общие настройки pf.
  set optimization normal
  set block-policy drop
  set loginterface $ext_if
  scrub on $ext_if all reassemble tcp
  # 3. QOS RULES (queueing)

  # Допустим, наш канал имеет пропускную способность 1 мегабит. Определим две очереди, первой отдадим 40% канала, второй - 60%, причем первая - приоритетнее. Директива borrow определяет возможность "одалживать" канал у соседней очереди, если та свободна. В первую очередь засунем именно себя :), во вторую - оставшихся пользователей.
  altq on $ext_if cbq bandwidth 1024Kb queue { toxa,
users }
  queue toxa bandwidth 40% priority 2 cbq(borrow)
  queue users bandwidth 60% priority 1 { deflt_users,
lan_users, im_users, http_users, mail_users }
  # Очередь для пользователей делится на четыре канала, которые отличаются приоритетом и шириной оставшегося канала. Очевидно, что пакеты от icq- или jabber-мессенджеров желательно доставлять в первую очередь.
  queue deflt_users bandwidth 30% priority 2
cbq(default borrow ecn)
  queue im_users bandwidth 20% priority 4 cbq(borrow
ecn)
  queue http_users bandwidth 30% priority 3 cbq(borrow ecn)
  queue mail_users bandwidth 20% priority 2 cbq(borrow ecn)
  # Далее разделим пакеты по очередям с помощью правил фильтрации.

  # 4. NAT/REDIRECTING RULES (translation)

  # NAT'им пользователей
  nat on $ext_if from $subnet to any -%26gt; $ext_if
  # Перебрасываем все www- и ftp-соединения пользователей на squid и ftp-proxy
  rdr pass on $int_if proto tcp from $subnet to ! $subnet
port { 80,8080} -%26gt; 127.0.0.1 port 3128
  rdr on $int_if proto tcp from $subnet to any port 21 -
%26gt; 127.0.0.1 port 8021

  # 5. PASS/BLOCK RULES (filtering)

  pass quick on $loop_if all
  # Политика по умолчанию "все запрещено"
  block quick log from any os NMAP
  block all
  # Прием против спуфинга
  block in quick on $ext_if from %26lt;priv_nets%26gt;
to any
  block out quick on $ext_if from any to
%26lt;priv_nets%26gt;
  antispoof for $ext_if
  pass in on $ext_if proto tcp from any to ($ext_if) port {
22,80 } modulate state
  pass in on $ext_if proto tcp from any to ($ext_if) port
%26gt; 49151 user proxy keep state
  pass in on $ext_if proto icmp from any to ($ext_if)
modulate state
  pass out on $ext_if from ($ext_if) to any modulate
state
  pass in quick on $int_if proto tcp from $adminbox to
$int_if flags S/SA modulate state
  pass out quick on $int_if proto tcp from $adminbox to
any flags S/SA modulate state queue toxa
  pass out quick on $int_if proto { udp, icmp } from
$adminbox to any modulate state queue toxa
  # Разруливаем пакеты по очередям
  block quick on $int_if proto tcp from $subnet to any
port { 135, 139, 445 }
  pass out on $int_if proto tcp from $subnet to any flags
S/SA modulate state queue deflt_users
  pass out on $int_if proto tcp from $subnet to any port
$im_ports flags S/SA modulate state queue im_users
  pass out on $int_if proto tcp from $subnet to any port
80 flags S/SA modulate state queue http_users
  pass out on $int_if proto tcp from $subnet to any port
{ 25,110 } flags S/SA modulate state queue mail_users
  pass out on $int_if proto { udp, icmp } from $subnet to
any modulate state queue deflt_users
```

Вот и все. Одна бессонная ночь - и непробиваемый шлюз для твоей домашней сети готов.

Правим pf.conf

Антон Карпов, toxa@cterra.ru

НЕПРИСТУПНЫЙ ПОЧТОВИК

ПОДНИМАЕМ БЕЗОПАСНУЮ И ФУНКЦИОНАЛЬНУЮ ПОЧТОВУЮ СИСТЕМУ

Чем сложнее система, чем больше в ней компонентов, тем она функциональнее, но в то же время тем сложнее следить за ее безопасностью. Тем не менее, сегодня мы будем строить максимально безопасный почтовый сервер, обслуживающий отдельный домен, и воспользуемся для этого проверенным временем qmail. А так как "просто почта" сегодня уже никому не нужна, будем учить qmail работать вместе с антивирусом ClamAV и лучшим на сегодняшний день убийцей спама SpamAssassin.

ОДА DJB

Стопроцентно безопасных программ не бывает. Но есть те, которые приблизились к этому показателю вплотную. Именно такие программы пишет американский профессор-математик Daniel Bernstein (http://cr.yp.to). Как и любой гений, он стоит в оппозиции по отношению ко многим вещам, и ему многое в этом мире не нравится. Например, ему не нравится то, что самый популярный dns-сервер BIND от Internet Systems Consortium, ставший стандартном де-факто для использования на UNIX-серверах, представляет собой весьма жалкое зрелище с точки зрения безопасности и дизайна. То же самое можно сказать о другом неписаном стандарте - почтовом сервере Sendmail. И хотя уже проходят те времена, когда эти ровесники интернета требовали исключительно права суперпользователя для работы, ожидая по свою душу очередного remote root эксплойта, хотя многое изменилось в лучшую сторону, стопроцентного доверия они так и не заслужили (и не заслужат: груз истории уязвимостей давит). Возможно, именно это побудило djb (так в интернете сокращенно зовут Бернштейна) написать свою реализацию dns-сервера под названием djbdns и smtp/pop3-сервера - qmail. Разумеется, qmail - не единственная безопасная альтернатива sendmail. Postfix - не уступающая по функциональности sendmail альтернатива от известного security-эксперта Wietse Venema. Но вот широко распространенных и безопасных, проверенных временем альтернатив BIND, кроме djbdns, похоже, не существует. Кроме того, программы djb многим не нравятся тем, что они разрушают годами складывавшиеся неписаные стандарты. Для чего? Не всегда "как есть" означает "как правильно".

ПЕРЕД ПОГРУЖЕНИЕМ

Было бы странно, если бы реформатор djb ограничился написанием сетевых демонов. Он решил начать с самого начала: с того, как и чем эти сервисы управляются. Представь, что ftp-сервер падает от DDoS-атаки. Пока администратор не придет и руками его не поднимет, сервис будет лежать. Чтобы избежать такой ситуации, было решено придумать "суперсервер", который бы осуществлял мониторинг запущенных подконтрольных ему сервисов и перезапускал их при необходимости. Подобной задачей, впрочем, занимается inetd, но он, как и многие продукты, не устраивал djb кривизной дизайна. Например, если падает сам inetd, то он тянет за собой все запущенные им сервисы. Для управления сервисами Бернштейн написал набор утилит под названием daemontools (http://cr.yp.to/daemontools.html). Если не вдаваться в подробности, то основные составные части daemontools - это монитор supervise, обработчик логов multilog и утилиты контроля сервисов svc/svstat. Помимо daemontools, им же был написан многофункциональный набор утилит ucspi-tcp (как раз прямой функциональный аналог inetd) для точного и детального контроля запускаемых демонов. Ключевая утилита из набора - tcpserver, которая запускает необходимый демон, устанавливает рабочее окружение (переменные среды) и контролирует все подключения к этому серверу, позволяя регулировать нагрузку, использование памяти и, если нужно, осуществлять контроль доступа.

В качестве операционной системы для построения высокопроизводительного надежного почтового сервера выберем FreeBSD 5.3 - лучшую серверную операционную систему на сегодняшний день ;). Детально процесс обновления системы до 5.3-STABLE описан в этом же номере, так что будем считать, что у тебя уже имеется свежая система с актуальным деревом портов. Поехали!

Первым делом ставим daemontools и ucspi-tcp.

```
# cd /usr/ports/sysutils/daemontools
# make install clean
# echo 'svscan_enable="YES"' %26gt;%26gt; /etc/rc.conf
# mkdir /var/service
```

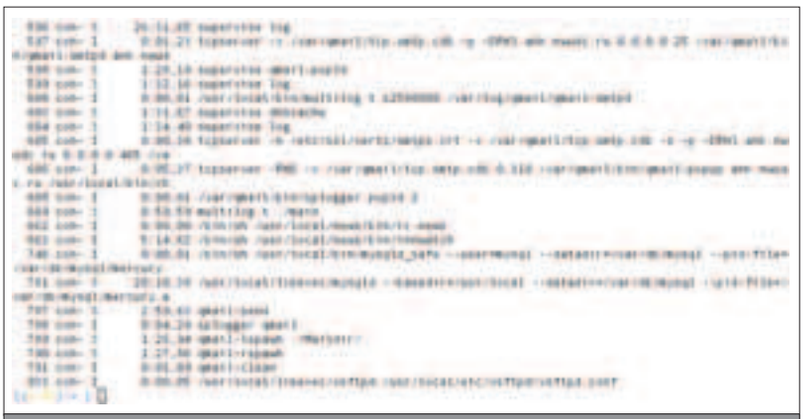
С ucspi-tcp придется немного повозиться. Дело в том, что для доступа к нашему будущему pop3/smtp-серверу с использованием безопасного ssl-соединения нужно пропатчить ucsp-tcp на предмет умения работы с SSL. По каким-то причинам этот патч отсутствует во FreeBSD-порте ucspi-tcp. Так что придется применить его самостоятельно.

```
# cd /usr/ports/sysutils/ucspi-tcp
# make patch
# wget http://www.nrg4u.com/qmail/ucspi-tcp-ssl-20020705.patch.gz
# gunzip ucspi-tcp-ssl-20020705.patch.gz
# cd work/ucspi-tcp-0.88
# patch %26lt; ../../ucspi-tcp-ssl-20020705.patch
# cd ../../
# make install clean
# rm ucspi-tcp-ssl-20020705.patch
```

ВАМ ПИСЬМО!

Пришло время ставить qmail в качестве pop3/smtp-сервера. Нас интересует порт с поддержкой SMTH-аутентификации и TLS.

Страница qmail



Процессы tcpserver

```
# cd /usr/ports/mail/qmail-smtp_auth+tls
# make WITH_QMAILQUEUE_PATCH=yes
WITH_BIG_TODO_PATCH=yes install clean
```

Авторизацию будем выполнять с помощью утилиты checkpassword.

```
# cd /usr/ports/security/checkpassword %26%26
make install clean
```

Затем сообщим системе, что у нас теперь qmail вместо sendmail:

```
# vi /etc/mail/mailer.conf

#sendmail /usr/libexec/sendmail/sendmail
#send-mail /usr/libexec/sendmail/sendmail
#mailq /usr/libexec/sendmail/sendmail
#newaliases /usr/libexec/sendmail/sendmail
#hoststat /usr/libexec/sendmail/sendmail
#purgestat /usr/libexec/sendmail/sendmail

sendmail /var/qmail/bin/sendmail
send-mail /var/qmail/bin/sendmail
mailq /var/qmail/bin/qmail-qread
newaliases /var/qmail/bin/newaliases
hoststat /var/qmail/bin/qmail-tcpto
purgestat /var/qmail/bin/qmail-tcpok

# echo 'sendmail_enable="NONE"' %26gt;%26gt;
/etc/rc.conf
```

Life With Qmail - обязательно к прочтению

Добавляем необходимых пользователей. Почти каждый процесс в qmail работает под отдельной учетной записью.

```
# pw groupadd nofiles
# pw groupadd qmail
# pw useradd alias -g nofiles -d /var/qmail/alias -s
/usr/sbin/nologin
# pw useradd qmaild -g nofiles -d /var/qmail -s
/usr/sbin/nologin
# pw useradd qmaill -g nofiles -d /var/qmail -s
/usr/sbin/nologin
# pw useradd qmailp -g nofiles -d /var/qmail -s
/usr/sbin/nologin
# pw useradd qmailq -g qmail -d /var/qmail -s
/usr/sbin/nologin
# pw useradd qmailr -g qmail -d /var/qmail -s
/usr/sbin/nologin
# pw useradd qmails -g qmail -d /var/qmail -s
/usr/sbin/nologin
```

Настало время настроить qmail. Делается это, как водится, командой echo ;).

```
# cd /var/qmail/control
# echo mail.mydomain.ru %26gt; me
# echo mydomain.ru %26gt; defaultdomain
# echo mydomain.ru %26gt; rcpthosts
# echo mail.mydomain.ru %26gt;%26gt; rcpthosts
# echo mydomain.ru %26gt; locals
# echo mail.mydomain.ru %26gt;%26gt; locals
```

Мы указали qmail его имя, то, на какие домены принимать почту и какие домены считать локальными. Затем создадим alias'ы на root'а и прочих пользователей, которым могут написать, но которые не смогут получать почту. Вся направляемая им почта будет скидываться пользователю toxa.

```
# cd /var/qmail/alias
# echo toxa %26gt; .qmail-root
# echo toxa %26gt; .qmail-postmaster
```

По умолчанию qmail использует формат почтового ящик Maildir, разработанный все тем же djb. В отличие от традиционного mbox, в котором вся почта складывается в один файл, Maildir хранит каждое письмо в отдельном файле.
По-моему, это вполне удобно, и нет нужды менять принятое по умолчанию поведение qmail. Создадим себе Maildir с помощью утилиты maildirmake:

```
# maildirmake /home/toxa/Maildir
```

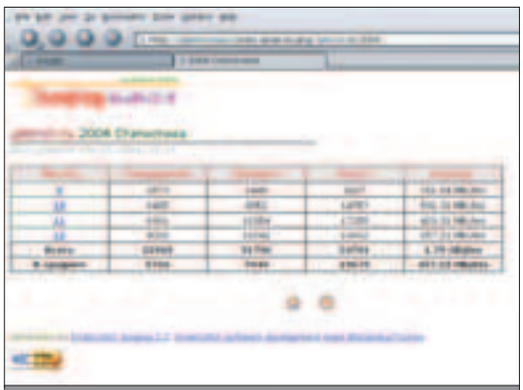
В появившемся каталоге Maildir увидишь три папки - cur, new, tmp. Новая почта сваливается в new. Так как сервер у нас будет доступен и по защищенному SSL-протоколу, сгенерируем себе самоподписанный сертификат для почты. Заметь, файл должен содержать как открытый ключ, так и секретный, так что с точки зрения x509 это не совсем сертификат (строго говоря, сертификат = открытый ключ + дополнительная информация):

```
# cd /tmp
# openssl genrsa -out mail.key 2048
# openssl req -new -key mail.key -out mail.csr
# openssl x509 -req -days 365 -in mail.csr -signkey
mail.key -out mail.crt
# cat mail.key %26gt;%26gt; mail.crt
# rm mail.key mail.csr
# mv mail.crt /etc/ssl/certs/
# chmod 600 /etc/ssl/certs/mail.crt
```

Запуск qmail будет осуществлять tcpserver из установленного нами набора ucspi-tcp. Мы запустим по два экземпляра pop3- и smtp-сервера, без SSL (на 110 и 25 портах соответственно) и с SSL (на 995 и 465 портах).
На первый взгляд, это выглядит громоздко, зато сразу видно четкое отделение мух от котлет, а защищенных соединений - от незащищенных. При этом не забудем, что smtp должен поддерживать авторизацию. Создаем каталог /var/qmail/runscripts, а в нем - подкаталоги qmail-pop3d, qmail-pop3ds, qmail-send, qmail-smtpd, qmail-smtpds. Если ты заметил, при запуске сервиса supervise'ом выполняется скрипт run в соответствующем каталоге. Здесь мы эти скрипты создадим руками и поправим под наши нужды.

```
# vi qmail-pop3d/run
#!/bin/sh
exec 2%26gt;%261
exec softlimit -m 5000000000 tcpserver -RHD -x
/var/qmail/tcp.smtp.cdb 0 110 \ /var/qmail/bin/qmail-
popup mail.myserver.ru /usr/local/bin/checkpassword \
/var/qmail/bin/qmail-pop3d Maildir
# vi qmail-pop3ds/run
#!/bin/sh
exec 2%26gt;%261
exec softlimit -m 5000000000 tcpserver -RHD -s -n
/etc/ssl/certs/mail.crt -x \ /var/qmail/tcp.smtp.cdb 0 995
/var/qmail/bin/qmail-popup mail.myserver.ru \
/usr/local/bin/checkpassword /var/qmail/bin/qmail-
pop3d Maildir
#vi qmail-smtpd/run
#!/bin/sh
exec 2%26gt;%261
exec envuidgid qmaild softlimit -d 30000000 tcpserv-
er -x /var/qmail/tcp.smtp.cdb \
-p -DRHI mail.myserver.ru 0.0.0.0 25
/var/qmail/bin/qmail-smtpd \
mail.myserver.ru /usr/local/bin/checkpassword
/usr/bin/true 2%26gt;%261
# vi qmail-smtpds/run
```

»



Isoqlog - генератор наглядной статистики по логам qmail

```
#!/bin/sh
exec 2%26gt;%261
exec envuidgid qmaild softlimit -m 30000000 tcpserver -n /etc/ssl/certs/mail.crt \
-x /var/qmail/tcp.smtp.cdb -s -p -DRHI mail.myserver.ru 0.0.0.0 465 \
/var/qmail/bin/qmail-smtpd mail.myserver.ru /usr/local/bin/checkpassword /usr/bin/true 2%26gt;%261
# vi qmail-send/run
#!/bin/sh
exec /var/qmail/rc
```

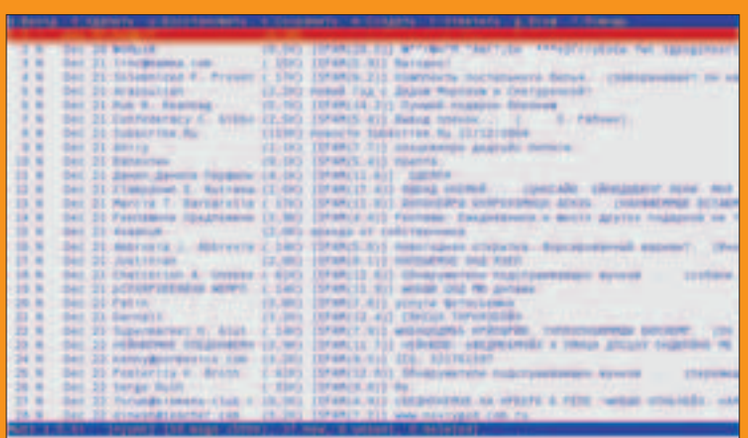
Как видно, цепь этих скриптов - установить переменные окружения и запустить tcpserver, который, в свою очередь, запускает соответствующий бинарник qmail-* с аргументами. В случае использования SSL мы указали путь к сертификату, в скриптах запуска smtpd/smtpds прописали checkpassword - это и есть поддержка smth auth в qmail. Заметь, что tcpserver также контролирует доступ согласно /var/qmail/tcp.smtp.cdb. Займемся этим файлом позже.

СПАМУ И ВИРУСАМ - БОЙ!

Без антиспам-системы сегодня не обходится ни один приличный почтовик. То же самое можно сказать и об антивирусном контроле проходящей через сервер корреспонденции. Потому мы призовем на помощь лучшие силы OpenSource в этой области - SpamAssassin и ClamAV AntiVirus.

Вот такой он, секретный ключ

Существует два принципиально разных подхода к фильтрации спама: так называемые server side и client side. Фильтрация на стороне сервера направлена на то, чтобы просто не принимать соединения от машин, которые рассылают спам. Собственно, она и основывается на технологиях определения "доверия" удаленным серверам, таким как Greylisting и RBL Checking (Blacklist checking). Само же письмо при этом, разумеется, никоим образом не затрагивается. Фильтрация на стороне клиента основана на определении вероятности того, что полученное письмо является спамом. Для этого применяются сложные статистические, лингвистические и эмпирические тесты, например, Bayes analysis.



Спам реально достал

Задача сервера - проанализировать и пометить проверенное письмо как "спам" или "не спам" (например, добавить в заголовок письма слово SPAM). Клиент все равно получит это сообщение, однако он вполне может настроить фильтрацию на своем почтовом клиенте и, например, складывать спам в trash не читая его. Или спам может просто убиваться на сервере по таким же признакам. У обоих методов есть свои преимущества и недостатки. В первом случае можно потерять важные сообщения при ложном срабатывании, во втором - придется принимать спам и хранить его на сервере, что в большой сети (с учетом того, что доля спама в наши дни в интернете достигает 80%) может оказаться критической. Плюс ко всему настойка анализатора на отлов всего спама и отсутствие ложных срабатываний требует времени. Как всегда, выход - золотая середина. Пожалуй, самый эффективный метод на сегодня - комбинация Greylisting и байесовской фильтрации.

```
# cd /usr/ports/mail/p5-Mail-SpamAssassin
# make install clean
# echo 'spamd_enable="YES"' %26gt;%26gt; /etc/rc.conf
# /usr/local/etc/rc.d/sa-spamd.sh start
```

Внесем в конфигурационный файл необходимые изменения:

```
# vi /usr/local/etc/mail/spamassassin/local.cf

# SpamAssassin ставит каждому письму "оценку" - количество баллов по шкале
# "спамовероятности". Например, оценка 50 - однозначно не спам, 0 или 2 - скорее всего не спам, 99
#- однозначно спам. Ограничиваем планку, выше которой все будет считаться спамом,
#шестью баллами. Подбери оптимальное значение для твоей сети на основе
#статистических данных (если есть ложные срабатывания - уменьши значения; если
#наоборот, то есть если спам проскакивает безнаказанно - увеличь).

required_hits 6.0
ok_languages en ru uk
ok_locales en ru
# В случае спама будем добавлять в заголовок письма уведомление и количество баллов.
rewrite_header Subject **[SPAM](_SCORE_)**
# Спам будет пересылаться вложением в уведомление почтовой системы.
report_safe 0
# включать статистику в заголовки письма
report_header 1
# Делаем систему самообучаемой
use_bayes 1
auto_learn 1
skip_rbl_checks 0
```

МНЕНИЕ ЭКСПЕРТА

■ Андрей Матвеев, редактор рубрик "Юниксоид" журнала "Хакер" (andrushock@real.xakep.ru)

В наши дни все больше внимания уделяют обеспечению комплексной безопасности систем электронной почты. Различные организации создают целые проекты по поддержанию так называемых "черных" списков, в которые занесены адреса отправителей, занимающихся массовой рассылкой информации рекламного характера. Разработчики программного обеспечения постоянно совершенствуют спам-фильтры, выпускают новые версии почтовых пользовательских агентов (MUA) и серверов SMTP/POP3/IMAP4, обладающих встроенной поддержкой аутентификации клиентов и шифрования передаваемых данных. Изо дня в день антивирусные компании трудятся не покладая рук над обновлениями к своим продуктам. Но, как показывает практика, велика вероятность того, что мы можем стать заложниками собственных средств защиты, так как добрый процент корреспонденции будет просто застревать в нашем же каскаде RBL-листов, почтовых фильтров, хэшированных базах доступа и, соответственно, не доходить до получателя. Поэтому тут главное не переборщить с защитой и выработать грамотную политику (к примеру, "что делать со спамом?" - удалять немедленно или доставлять клиенту с модифицированной темой письма).

Проверить работу ssl-"обвертки" можно утилитой openssl s_client

```
# По умолчанию spamd, обучаясь, складывает токены в $HOME пользователя. Чтобы
# иметь общую базу на всех, будем складывать их в одном месте.
bayes_path /usr/local/etc/mail/spamassassin/
```

Протестируем работу spamd натравив на него какое-нибудь письмо:

```
# cat message | spamc
```

На stdout будет выведено письмо со статистической информацией в заголовках:

```
X-Spam-Status: No, score=-99.4 required=4.0 tests=AWL,BAYES_50,
RCVD_DOUBLE_IP_LOOSE,SUBJ_ILLEGAL_CHARS,USER_IN_WHITELIST
autolearn=no version=3.0.1
X-Spam-Checker-Version: SpamAssassin 3.0.1 (2004-10-22) on mail.myserver.ru
```

Ставим clamav:

```
# cd /usr/ports/security/clamav
# make install clean
# echo 'clamav_clamd_enable="YES"' %26gt;%26gt; /etc/rc.conf
# echo 'clamav_freshclam_enable="YES"' %26gt;%26gt; /etc/rc.conf
# /usr/local/etc/rc.d/clamav-clamd.sh start
# /usr/local/etc/rc.d/clamav-freshclam.sh start
```

Freshclam будет регулярно обновлять базу штаммов вирусов, clamd будет непосредственно отвечать за анализ файлов.

ОБЪЕДИНЯЕМ ВСЕ ВОЕДИНО

■ Итак, у нас стоит qmail (еще не запущенный), clamav и spamassassin. Пока они ничего друг о друге не знают, и чтобы превратить разрозненные компоненты в мощный почтовый монолит, мы воспользуемся маленькой и шустрой программкой simscan. На момент написания статьи ее не было в портах, так что нам придется ставить ее руками, предварительно добавив необходимые программы:

```
# cd /usr/ports/mail/ripmime
# make install clean
# wget http://www.inter7.com/simscan/simscan-1.0.8.tar.gz
# tar xzf simscan-1.0.8.tar.gz %26%26 cd simscan-1.0.8
# ./configure --enable-spam=y --enable-received=y --enable-ripmime --enable-spamassassin-path=/usr/local/bin/spamassassin --enable-clamavdb-path=/usr/local/share/clamav --enable-sigtool-path=/usr/local/bin/sigtool --enable-attach=y --enable-spam-passthru=y --enable-quarantinedir=/var/qmail/simscan/quarantined
# make %26%26 make install
```

Подробнее про все опции читай в README. Принцип работы simscan заключается в подмене оригинальной программы процессинга почтовой очереди /var/qmail/bin/qmail-queue на /var/mail/bin/simscan, которая умеет обращаться к spamd/clamd, а в остальном ведет себя так же, как и qmail-queue. Вот тут мы и вернемся к файлу /var/qmail/tcp.smtp.cdb. Создай в /var/qmail файл следующего содержания:

```
127.0.0.1:allow,RELAYCLIENT=""
192.168.0.:allow,RELAYCLIENT="",QMAILQUEUE="/var/qmail/bin/simscan"
:allow,QMAILQUEUE="/var/qmail/bin/simscan"
```

Затем преврати его в cdb:

```
# tcprules tcp.smtp.cdb tcp.smtp.cdb.tmp %26lt; tcp.smtp
```

В этом файле устанавливаем переменные окружения для хостов. RELAYCLIENT позволяет указанному хосту использовать qmail для пересылки почты хостам, не указанным в rcpthosts, то есть использовать сервер как relay. Мы разрешили это локальному хосту и нашей подсети 192.168.0.0/24. Очевидно, что все остальные будут уметь доставлять почту только тем хостам, которые обслуживает qmail (в нашем случае это mail.mydomain.ru). Таким образом, promisc relay исключен. Как же клиенты будут отправлять почту, например, из дома? Для этого мы и включили smtp-авторизацию, а упомянутый выше checkpassword именно тем и занимается, что после проверки пароля выставляет для сессии переменную RELAYCLIENT, позволяя пересылать почту. Также для всех хостов, кроме локального, мы переписали переменную QMAILQUEUE, чтобы она указывала на simscan. Вот теперь запускаем все четыре экземпляра qmail:

```
# ln -s /var/qmail/runscripts/qmail-pop3d /var/service
# ln -s /var/qmail/runscripts/qmail-pop3ds /var/service
# ln -s /var/qmail/runscripts/qmail-smtpd /var/service
# ln -s /var/qmail/runscripts/qmail-smtpds /var/service
# ln -s /var/qmail/runscripts/qmail-send /var/service
```

Проверяем работу системы, посылая почту из локальной сети и из интернета, убеждаемся, что без smth auth снаружи не пускает, смотрим логи clamd/spamd. Если все работает, радуемся и запускаем систему в эксплуатацию. Если нет - идем на www.google.com и учимся пользоваться поиском :).

Докучаев Дмитрий aka Forb (forb@real.xakep.ru)

ПОСТРОЙ СВОЙ ДОМЕН

ПОДНЯТИЕ ГЛАВНОГО КОНТРОЛЛЕРА ДОМЕНА В *NIX

Тебе удалось устроиться на работу системным администратором в крутую фирму? Мои поздравления! Конечно, ты уже поставил Linux в качестве главного сервера и подумываешь над организацией корпоративного домена. Ты идешь в правильном направлении. Если тщательно настроить контроллер, то он будет работать ничуть не хуже ActiveDirectory. Обо всем по порядку.

ЧТО ТАКОЕ ДОМЕН?

Любой уважающий себя системный администратор должен знать основные сетевые термины. Домен - это своеобразное объединение нескольких машин в один узел. Естественно, такая цепочка компьютеров нуждается в управлении. Для этого создается главный контроллер домена (Primary Domain Controller, или PDC). Организация сетевого домена имеет как минимум три преимущества.

❶. На контроллере будут храниться все доменные аккаунты, под которыми можно логиниться на любой машине, входящей в домен. Таким образом, тебе не нужно забивать сотню пользователей на каждом компьютере. Достаточно создать базу на сервере.

❷. На PDC сохраняются все индивидуальные настройки (профили) пользователей, а также может быть выделено место на сетевом диске для хранения данных. Что касается профилей, то они автоматически перемещаются на клиентскую часть при входе и на PDC при выходе. То есть профили в домене постоянно синхронизируются. Таким образом, пользователь, залогинившись на любой машине в домене, может не озадачиваться перенастройкой системы.

❸. В домене можно легко организовать собственный сервер печати. После такого нововведения никто никому не запретит распечатать документ на сетевом принтере.

На самом деле количество преимуществ больше трех. Ты это поймешь, когда дочитаешь материал до конца. Но за удовольствие надо платить, поэтому, прежде чем твой PDC заработает, нужно пройти семь кругов ада, чтобы успешно его настроить :). Я, конечно, утрирую, но доля правды в этом высказывании есть. С твоего позволения, я начинаю рассказ о правильной настройке Primary Domain Controller.

УСТАНОВКА БАЗОВОГО СОФТА

Прежде чем настраивать контроллер, необходимо позаботиться о программном обеспечении. Каждому юниксоиду известно, что ничего лучше Samba в этом плане еще не придумано. Существует две ветки этого продукта - вторая и третья. Судя по высказываниям администраторов, третья веточка пока еще сырая, поэтому мы организуем контроллер на стабильной версии 2.2.12. Скачивай этот релиз с главного сайта по ссылке http://download.samba.org/samba/ftp и приступай к сборке демона. Если у тебя имеется родной RPM, можешь поставить его - бинарники испечены со всеми нужными параметрами.

При нормальных условиях Samba соберется без приключений. Для упорядоченности можешь указать опции --prefix=/usr и --sysconfdir=/etc/samba. В этом случае все конфигурационные файлы будут скопированы в /etc/samba, а сам демон определится в каталоге /usr/sbin.

ОТЛАДКА КОНФИГУРАЦИИ

Вот, собственно, и вся установка :). Просто, не правда ли? На самом деле инсталляция и настройка - это принципиально разные вещи, и, как правило, на отладку конфигурационных файлов уходит гораздо больше времени, чем на первоначальную сборку программы. Для тебя главное - вникнуть в смысл каждой конфигурационной директивы и сделать из громоздкого примера компактный файл. Чем сейчас и займемся.

Зайди в директорию /etc/samba и открой файл smb.conf. В самом начале ты увидишь главную секцию [Global]. В ней описываются важные параметры, от которых зависит работа демона. Первые две директивы называются workgroup и netbios name соответственно. Значение workgroup определяет название домена. Можешь не заморачиваться с выбором имени - обзови домен легко запоминающимся словом (названием твоей фирмы, например). Параметр netbios name отвечает за имя PDC в домене. К примеру, если ты определишь эту директиву как server, то клиенты смогут использовать расшаренные ресурсы и подключать сетевые диски обращаясь к пути "\\server".

Далее идет параметр server string. Это не что иное, как описание сервера. Пользы от введения этой директивы мало, однако комментарий может облегчить жизнь твоим некомпетентным коллегам. В этом случае описание "Самые нужные программы и go-

```
[root@tim samba-2.2.9]# ls
COPYING  examples  packaging  Read-Manifest-Now  REVISION  source  testsuite
docs     Manifest  pop        README             Roadmap   swat    WHATSNEW.txt
[root@tim samba-2.2.9]# cd source/
[root@tim source]# ./configure --prefix=/usr --sysconfdir=/etc
creating cache ./config.cache
checking for gcc... gcc
checking whether the C compiler (gcc -O  ) works... yes
checking whether the C compiler (gcc -O  ) is a cross-compiler... no
checking whether we are using GNU C... yes
checking whether gcc accepts -g... yes
checking for a BSD compatible install... /usr/bin/ginstall -c
checking for gawk... gawk
checking if the linker (ld) is GNU ld... yes
checking for passwd... /usr/bin/passwd
checking whether gcc and cc understand -c and -o together... yes
checking that the C compiler understands volatile... yes
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu
checking build system type... i686-pc-linux-gnu
checking config.cache system type... ansi
checking for LFS support... yes
checking for inline... inline
checking how to run the C preprocessor... gcc -E
```

Конфигурируем Samba

кументы" является наиболее правильным :). Хотя решать, как говорится, тебе.

После определения имен и описаний, ты наткнешься на ряд параметров, которые нужны для организации журналирования и безопасности. Для правильного логгинга имеются опции log file, log level и max log size. При первоначальной настройке и обкатке PDC оставь название лога в виде /var/log/samba/smbd.%m. В этом случае ты всегда можешь разобраться в проблеме, если какая-то машина не захочет входить в сетевой домен. Размер лога определяет максимальный объем журнала (в килобайтах). Если обнулить значение этой директивы, размер контролироваться не будет. log level определяет уровень логирования. Чем он выше – тем больше информации запишется в журнал. Значение 5 является оптимальным. Ставить его выше рекомендуется лишь в том случае, если возникают какие-то проблемы. Впрочем, не каждый администратор разберется в записях, сделанных при высоком уровне логирования. Следующая порция опций - hosts allow, security, encrypt passwords, smb passwd file, null passwords и pam password change. Первая директива определяет список хостов, с которых разрешено логиниться в домен. Разумнее всего внести в него диапазон IP-адресов твоей локальной сети. Параметр Security определяет назначение сервера PDC. Если ты задашь его значение строкой user, это будет означать, что все доменные аккаунты находятся на этом же сервере. Обязательно включи encrypt passwords, иначе все пароли будут передаваться открытым текстом, что очень небезопасно. Null passwords разумнее всего отключить из соображений безопасности (если, конечно, сотрудники не пользуются общей учетной записью без заданного пароля). И, наконец, последний параметр в моем списке определяет возможность смены пароля. Если его значение равно yes, то любой желающий сможет изменить пароль нажатием <CTRL>+<ALT>+<DEL> на клиентской машине.

Все параметры, оканчивающиеся на *master, должны иметь значение yes. Именно они указывают Samba, чтобы она выступала в роли главного контроллера домена. Включи также директивы wins support, domain logons и nt acl support. Первая опция заставляет smbd резолвить netbios-имена, а последняя реализует копирование UNIX-прав на клиентскую машину.

Остальные опции можно оставить установленными по умолчанию. Теперь самое время обратить внимание на пути к каталогу с профилями и домашними папками. За это дело отвечают две директивы:

```
logon drive = Z:
logon path = \\netbios name\profiles\%u.
```

Первая запись определяет имя сетевого диска, который будет автоматически (или вручную) подключаться при заходе в домен. Вторая задает путь к каталогу, где хранится профиль клиента. Значение переменной %u становится равным имени пользователя, а netbios name опять же должно замениться на netbios имя PDC.

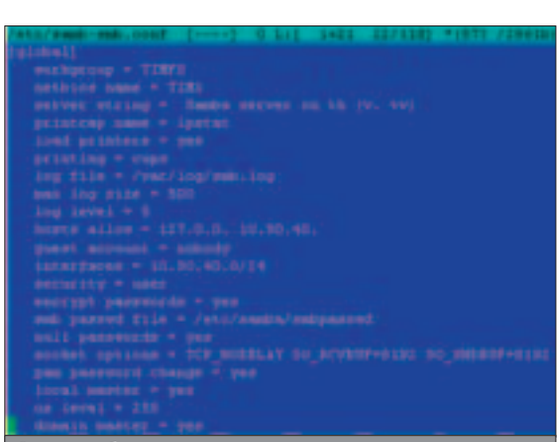
На этом с настройкой глобальной сети покончено. Но, к сожалению, одной секцией не обойтись. Прежде чем обкатывать функциональность домена, нужно задать список расшаренных ресурсов. Некоторые из них являются обязательными, остальные оформляются в произвольном стиле. Название ресурса помещается в квадратных скобках. Ниже, как ты, наверное, догадался, располагается его описание в виде ряда параметров.

К обязательным шарам относятся ресурсы Netlogon, Profiles и Homes. Из ресурса Netlogon будет взят сценарий подключения, который ранее определялся в директиве logon script. Profiles предназначается для того, чтобы объявить демону локальный путь, откуда берется пользовательский профиль. И, наконец, ресурс Homes отвечает за местоположения индивидуальных данных пользователя. Именно содержимое \\PDC\homes подключается к сетевому диску, имя которого определено в logon drive.

Остальные сетевые ресурсы имеют произвольное имя и определяются по »

Если у тебя не получилось отконфигурировать Samba, попробуй использовать swat. Этот сервис есть в каждом дистрибутиве и существенно облегчает настройку PDC.

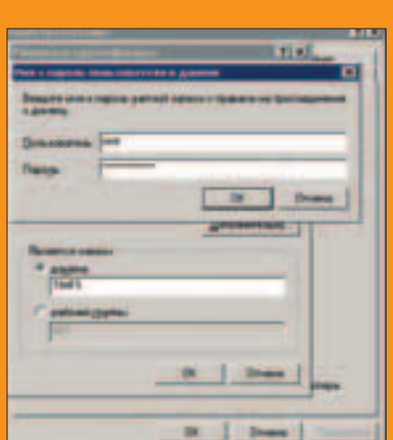
Чтобы пользователь при входе мог видеть определенные сетевые диски, создай стартовый сценарий logon.bat в шаре netlogon.



Самая глобальная вкладка

ОБСЛУЖИМ КЛИЕНТА

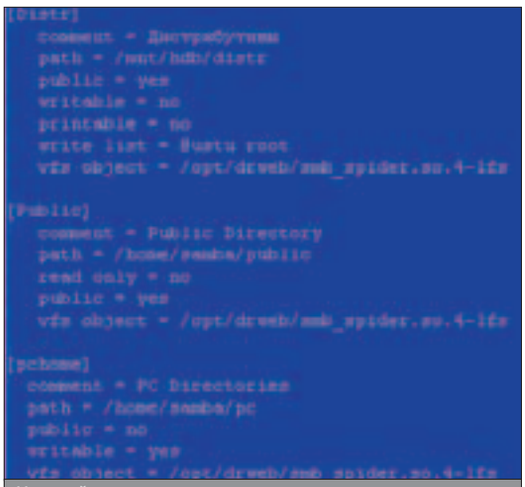
Для того чтобы сотрудники подразделения могли использовать функциональность домена, необходимо завести в него ряд рабочих машин. Если на клиентском компьютере установлена Win2k, проблем почти не будет: заходишь в свойства "Мой компьютер", выбираешь вкладку "Сетевая идентификация" и набираешь имя домена. Windows попросит у тебя логин и пароль. В качестве логина укажи ключевое слово root, а пароль пиши тот, который задавал при оформлении администраторского аккаунта. После финальной перезагрузки можно логиниться под сетевым именем и работать в домене. Все шары находятся в "Сетевом окружении" в рабочей группе netbios name (эта директива находится в самом начале smb.conf).



Добро пожаловать в домен!

В случае с WinXP все немного сложнее. Прежде чем заносить машину в домен, надо чуток подправить локальную политику безопасности. В "Параметрах безопасности" отключи опцию "Требуется цифровая подпись для члена домена". Теперь можно пользоваться машиной :). Кстати, WinXP, в отличие от Win2k, требует пароль администратора не только для введения станции в домен, а также при ее исключении.

Если вдруг твои коллеги работают на машинах Win9x, то здесь вообще нет никаких проблем. Дело в том, что Win9x не нужно вводить в домен, соответственно не требуется заносить их сетевые имена в базу Samba. Достаточно изменить параметры "Клиента для сетей MicroSoft", указав в соответствующем поле имя домена.



Настройка расшаренных ресурсов

твоему желанию. Я не буду описывать опции, применяемые к настройке расшаренных ресурсов, - ты всегда сможешь найти их в справочной странице или взять из примера моего конфигурационного файла. Бери рабочий example.conf по ссылке http://kamensk.net.ru/forb/1/x/example_smb.conf.

Разобравшись с синтаксисом и смыслом каждого параметра, ты настроишь конфигурацию без особых трудностей. Но никто не застрахован от ошибок. Чтобы проверить правильность составления smb.conf, запусти утилиту testparm. Программа быстро найдет ошибки, а если их нет - радостно сообщит об этом.

ОФОРМЛЯЕМ ПОЛЬЗОВАТЕЛЕЙ И НАСТРАИВАЕМ КЛИЕНТОВ

Я думаю, не стоит напоминать о том, что все каталоги, упомянутые в smb.conf, должны реально существовать. Это понятно каждому. Настройка конфигурации плавно перетечет в процедуру создания пользовательской базы. Samba требует, чтобы имя пользователя записывалось как в базе smbpasswd, так и в обычном /etc/passwd. Это связано с тем, что демон выполняет setuid() на пользовательский идентификатор после того, как клиент войдет в сеть. Соответственно, если ты создашь сетевой аккаунт vasya, права на домашний каталог должны позволять Василию просматривать, закачивать и удалять новые файлы. Это достигается командой chmod vasya:smbusers /path/to/homes/vasya. Аналогичная команда должна быть выполнена для каталога с профилем Васи.

Советую написать скрипт, который будет добавлять пользователя (либо группу пользователей) в домен. Тем самым ты избавишь себя от рутинной работы.

Drwebd под Linux и модуль smb_spider ты можешь скачать с официального сайта http://download.drweb.com/unix

```
[root@tim samba]# testparm
Load smb config files from /etc/samba/smb.conf
Processing section "[homes]"
Processing section "[netlogon]"
Processing section "[Profiles]"
Processing section "[tmp]"
Processing section "[Fast]"
Processing section "[Distr]"
Processing section "[Public]"
Processing section "[pchome]"
Processing section "[Super]"
Loaded services file OK.
Press enter to see a dump of your service definitions
```
Проверка на вшивость

Создавать пользователя Васю, как я уже сказал, придется два раза. Первая команда является классической - useradd vasya -s /dev/null -d /home/users/vasya -g smbusers. Вторая чуть попроще: smbpasswd -a -n vasya. Опции к скрипту smbpasswd (именно он добавляет аккаунт в базу данных smbd) говорят о том, что заводится новый пользователь с пустым паролем. При желании можно указать произвольный пароль опуская опцию -n.

Помимо учетных записей клиентов, в базу нужно занести логин root. Да-да, root - он и в Африке root. Администратор под этим логином может пользоваться всеми ресурсами и заводить машины в домен. Соответственно, для абсолютной безопасности необходимо установить root длинный и сложный пароль (совсем не обязательно, чтобы он совпадал с системным).

Примечательно, что прежде чем завести машину в домен, необходимо прописать ее в базу данных smbpasswd. При этом имя машины будет совпадать с именем рабочей станции и оканчиваться значком доллара. К примеру, если в корпоративной сети есть компьютер с именем designer, аккаунт будет выглядеть как designer$. Чтобы smbpasswd понял, что имеет дело с клиентской записью, следует указывать дополнительный параметр -m. Таким образом, чтобы оформить компьютер дизайнера, выполняем две команды:

```
useradd designer$ -s /dev/null -d /dev/null -g workstations
smbpasswd -a -m designer$.
```

ПОСЛЕДНИЕ ШТРИХИ

На этой оптимистической ноте можно закончить с настройкой. Осталось лишь запустить демоны smbd и nmbd (smbd -vD; nmbd -vD или /etc/init.d/smbd start) и завести нужные машины в домен. После перезагрузки будет предложено ввести сетевой логин и пароль. Если все сделано правильно, после первого захода пользователя будет создан локальный профиль, который уже можно изменять. Когда пользователь завершит работу, увесистый профиль будет передан по сети и помещен в папку /home/profiles/имя_пользователя (или в другой каталог, оговоренный в smb.conf). После тестирования работы smbd можно чуть-чуть изменить конфигурацию, а именно: урезать объем журнала и понизить уровень логирования.

ВИРУСНАЯ АТАКА

Никто не застрахован от заражения вирусами. Особенно эта тема актуальна для администраторов домена. Только представь: случайно в расшаренный каталог был выложен какой-нибудь вирус. Все пользователи локальной сети бережно скачали и запустили хакерское творение на своих машинах. В итоге получается, что добрая половина компьютеров заражена опасным вирусом. Ты можешь утверждать, что в твоей фирме на машины установлены антивирусы, но это скорее исключение, чем правило. Даже если на клиентских компьютерах отсутствует антивирус, запретить передачу заразы по сети очень просто. Для этого тебе понадобится drwebd и модуль drweb-samba. Установка антивируса - тема отдельной статьи, поэтому тебе придется поставить drwebd самостоятельно. А про то, как интегрировать Samba и drweb-samba, расскажу с удовольствием.

Итак, мы имеем распакованный модуль smb_spider.so и запущенный drwebd. Чтобы заставить антивирус-

> Никто не застрахован от заражения вирусами.

```
[root@tim samba]# useradd vasya -s /dev/null -d /home/samba/users/vasya -g stude
nt
[root@tim samba]# /usr/bin/passwd -l vasya
usage: passwd [-k] [-k] ["username"]
       -h          - print this text and exit;
       -k          - keep non-expired authentication tokens;
       "username"  - (superuser may) update tokens for named user.
[root@tim samba]# /usr/sbin/passwd -l vasya
[root@tim samba]# smbpasswd -a vasya
New SMB password:
Retype new SMB password:
Added user vasya.
[root@tim samba]# id vasya
uid=1032(vasya) gid=506(student)
[root@tim samba]# grep vasya /etc/samba/smbpasswd
vasya:1032:5D1072C04445E010AA33B4358514D4EE:AE059F7DC61A5C6481A06D0C6DD44C448:[U
          ]:LCT-4186F216:
[root@tim samba]# useradd vasya1 -s /dev/null -d /home/samba/pc -g workstation
[root@tim samba]# /usr/sbin/passwd -l vasya1
[root@tim samba]# smbpasswd -a -m vasya1
Added user vasya1.
[root@tim samba]#
```
Добавляем пользователя и его машину

ную программу проверять все файлы "на лету", необходимо настроить конфигурационный файл модуля, а также чуток подправить smb.conf. Начнем с настройки spider'а.

Большинство параметров в /etc/drweb/smb_spider.conf вообще не стоит изменять. Следует лишь выбрать метод сканирования. Их может быть три: проверка файла при любом доступе к нему, когда осуществляется заливка, или скачивание. По умолчанию указан первый способ. Директива имеет название ScanMode.

Чуть ниже находится описание действий при наличии вируса (параметр Infected). Возможные варианты: вылечить файл, переименовать файл и запретить доступ к нему, удалить файл, отправить вирус в карантин или вообще ничего не предпринимать :). Наиболее приемлемыми решениями являются удаление файла и лечение. Однако далеко не все вирусы лечит drweb. В связи с этим имеется описание еще одного события (Incureable), которое обрабатывается, если зараженный файл не может быть излечен. Тут разумнее всего запретить доступ к файлу, удалить заразу или... отправить на карантин. Далее следует секция настройки оповещения. Здесь все понятно: достаточно задать адрес SMTP-сервера и твой e-mail. Как только Spider увидит вирус, тебе свалится тревожное письмецо.

Теперь самое время настроить smb.conf. Прежде чем открывать его для редактирования, сделай символическую ссылку с именем smb_spider.so на файл smb_spider.so.4. Именно этот модуль используется для работы с последними версиями smbd. После линковки файлов открывай smb.conf и в конце описания расшаренного ресурса добавь следующий параметр:

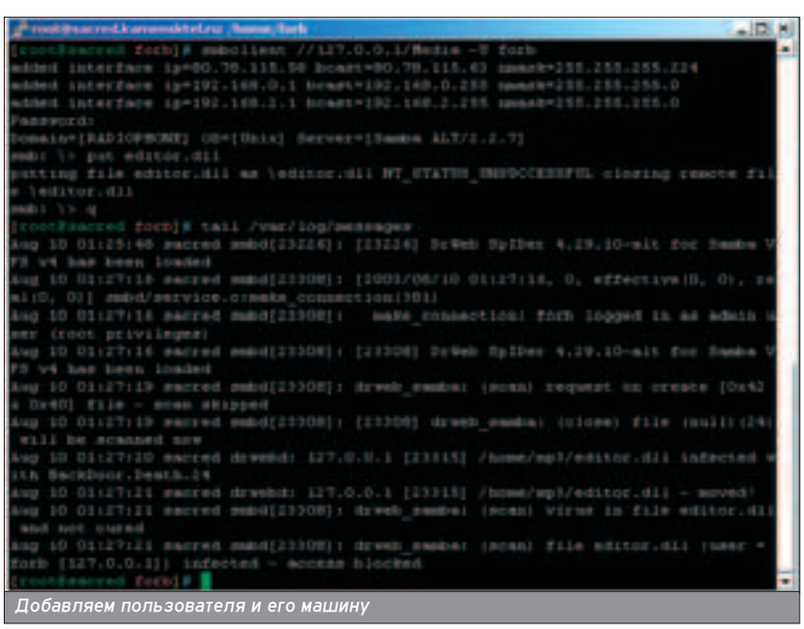
```
vfs object = /opt/drweb/smb_spider.so.
```

Какие шары выбрать - решать тебе. Напомню, что если ты воткнешь проверку для ресурса Profiles, это сильно замедлит работу (передача профиля по сети и без проверки на вирусы занимает много времени), но зато все персональные файлы пользователя, включая содержимое папки "Мои документы", будут проверены. Разумным решением является проверка ресурса Homes, а также всех шаров с выложенными документами и программным обеспечением. В общем, здесь все зависит от твоей фантазии.

После пластической операции над conf-файлом выполни команду /etc/init.d/smbd reload и проверь работу Spider. Для проверки зайди на какой-нибудь хакерский сайт и стяни оттуда новомодный троянец. Затем попытайся залить его в расшаренную папку. При правильной настройке drwebd быстро опознает вирус, и модуль тут же запретит доступ к небезопасному файлу.

ВСЕМ СПАСИБО, ВСЕ СВОБОДНЫ

Если ты читаешь эти строки и хоть что-нибудь понял в вышеизложенном тексте, то можешь с уверенностью сказать, что твой PDC будет исправно обслуживать домен долгие годы. Но не забывай своевременно обновлять версию Samba и модуля Spider. Я сознательно не написал о том, как настраивать сетевой принтер. Для этого дела существуют свои приемы и директивы. Почитай man на smb.conf и все поймешь сам. Если же вдруг ты наткнешься на проблему или у тебя появится неотложный вопрос - пиши письмо, разберемся за умеренную плату :). Я верю, что после поднятия сетевого домена твой шеф будет носить тебя на руках, коллеги сразу же проникнутся уважением, а зарплата стремительно поползет вверх :). А теперь бросай все дела и дуй на download.samba.org за свежей версией Samba, которая сделает твою работу легче и приятнее.



Добавляем пользователя и его машину

Антон Карпов, toxa@cterra.ru

БРОНЕБОЙНЫЙ DNS

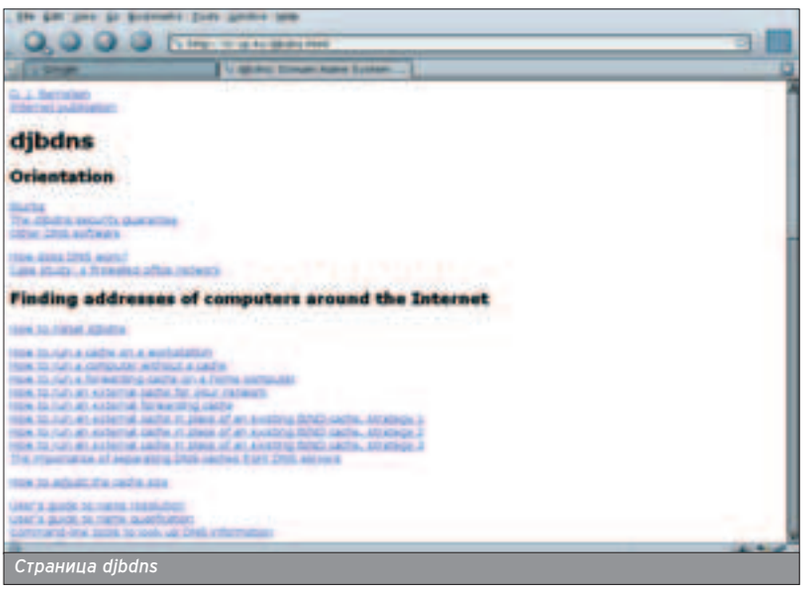
ПОДНИМАЕМ БЕЗОПАСНЫЙ И ФУНКЦИОНАЛЬНЫЙ DNS-СЕРВЕР

Ты уже прочитал (или еще прочтешь) статью о настройке почтового сервера в этом номере. Как ты уже, наверное, догадался, почте, как и другим сетевым сервисам, жить без DNS проблематично. Доменная система имен - основа всей сети, поэтому обустроить свой "именной" уголок безопасно и корректно - одна из основных задач администратора.

DJBDNS - ТРИ В ОДНОМ

Знакомься: djbdns - полноценная реализация для работы с DNS от Дэна Бернштейна. Как нельзя лучше подходит для решения именно этой проблемы. Итак, настройки требует DNS-сервер, который предоставлял бы внешним машинам информацию о нашей зоне, а клиентам Сети - информацию об адресах машин в интернете. Если ты имел дело с BIND, то знаешь, что он "и швец, и жнец, и на дуде игрец", то есть в зависимости от настроек может и держать зону, и играть роль кеширующего сервера/форвардера, и отдавать зону slave-серверам. Это не самое элегантное решение, что доказывает богатая история уязвимостей BIND и постоянные проблемы с его корректной настройкой у начинающих администраторов. Как же должно быть? Так, как в пакете djbdns, который представляет собой набор из трех основных программ: tinydns (не умеет ничего кроме как обслуживать зоны); dnscache (отвечает лишь за кеширование и разрешение внешних имен); axfrdns (занимается отдачей зон tinydns'a slave-серверам, точнее named'ам, так как для распространения зоны между двумя tinydns djb предлагает другие механизмы). Почему все так сложно? Эти три программы функционируют независимо друг от друга, и не существует даже теоретической возможности, например, некорректно настроить DNS-сервер так, чтобы он отвечал на рекурсивные запросы (такие сервера часто используются для DDoS-атак), или удаленно получить контроль над кешем (доступный для соединений извне tinydns не работает с кешем).

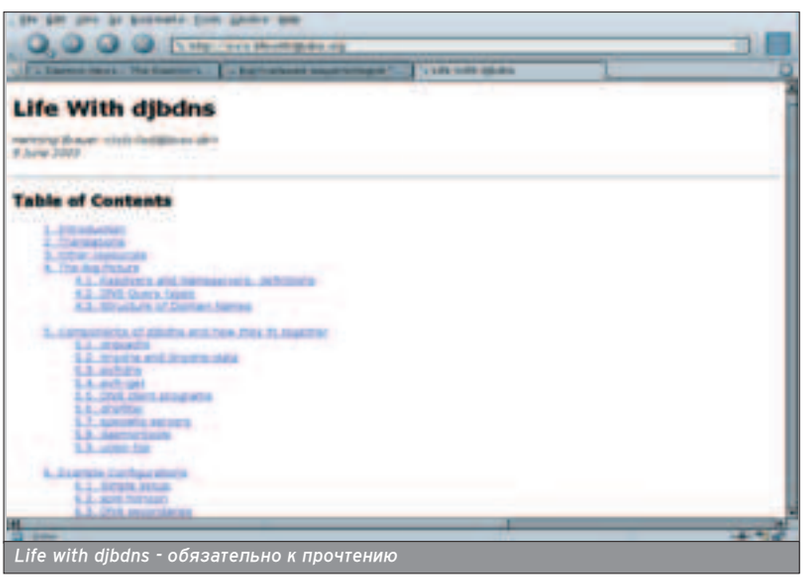
Будем считать, что у нас в распоряжении вторичный DNS-сервер BIND. Разгуляемся по полной настроив dnscache, tinynds и axfrdns. Если ты когда-нибудь имел дело с BIND, то постарайся забыть все, что ты знал о нем :). Если нет - тем лучше для твоей психики: named, как правило, слушает



Страница djbdns

на всех доступных интерфейсах, принимая запросы на внутреннем интерфейсе от клиентов для разрешения имен внешних машин, а на внешнем интерфейсе - от серверов для предоставления информации о машинах в своей зоне и трансфера зоны вторичным серверам. Разбивая систему на функциональности, получаем следующую картину: на внутренний интерфейс приходят исключительно пользовательские запросы. Значит, на него нужно повесить dnscache, который будет резолвить адреса внешних машин и кешировать результат.

На внешний интерфейс приходят запросы к нашей зоне как на разрешение имен, так и на трансфер. Значит, на него повесим tinydns, который понятия не имеет, что такое рекурсив-



Life with djbdns - обязательно к прочтению

ные запросы, и отвечает только на запросы своей зоны, и axfrdns, который будет отдавать зону определенным slave-серверам. В качестве ОС для построения DNS-сервера избираем FreeBSD. Но о вкусах не спорят. Все нижеизложенные инструкции легко можно применить и к Linux/OpenBSD/NetBSD и т.д.

```
cd /usr/ports/dns/djbdns %26%26 make install clean
```

Добавляем пользователей для работы dnscache, tinydns и axfrdns, так как в целях безопасности каждый из демонов запускается под своей учетной записью. Обработчик логов запускается под отдельной учетной записью dnslog:

```
# pw adduser -d /dev/null -s /usr/sbin/nologin -c
"DJBDNS dnscache user" dnscache
# pw adduser -d /dev/null -s /usr/sbin/nologin -c
"DJBDNS tinydns user" tinydns
# pw adduser -d /dev/null -s /usr/sbin/nologin -c
"DJBDNS axfrdns user" axfrdns
# pw adduser -d /dev/null -s /usr/sbin/nologin -c
"DJBDNS dnslog user" dnslog
```

ПОРЯДОК ДЕЛЕГИРОВАНИЯ ЗОНЫ

■ Очень важно помнить последовательность добавления вторичных серверов и не совершать популярной ошибки под названием lame delegation. Правильный порядок действия таков.

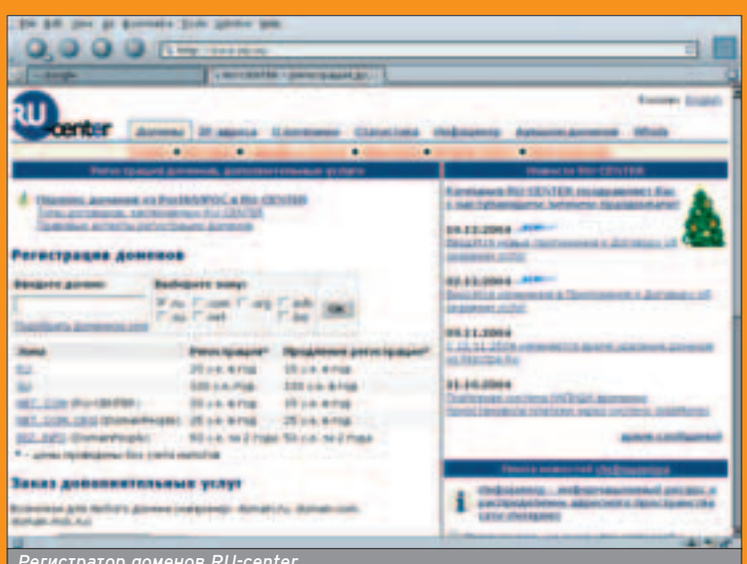
❶. Настроить вторичный dns-сервер на получение зоны с первичного.

❼. Разрешить на первичном сервере трансфер зоны вторичным dns-сервером. В случае BIND эта процедура включает в себя правку конфигурационных фалов named.conf на обоих серверах, открытие портов 53/tcp на соответствующие хосты, просмотр логов named на наличие ошибок и т.д. В случае tinydns эта процедура сводится к прописыванию одной строчки в Makefile. Если что не так, rsync тут же выдаст на консоль диагностические ошибки.

❽. Добавить в конфигурационный файл зоны первичного сервера NS-запись, указывающую на новый вторичный сервер.

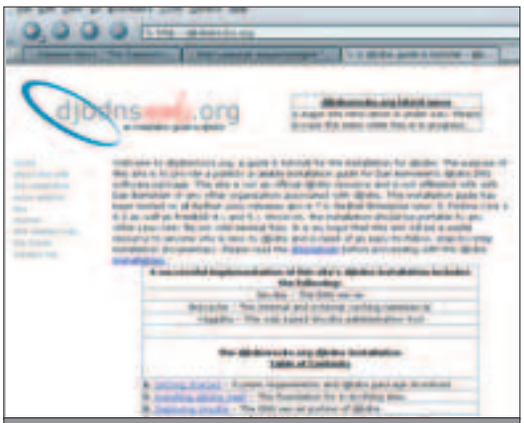
❿. Если новый вторичный сервер входит в обслуживаемую зону, добавить соответствующую A-запись.

❺. Сообщить вышестоящему регистратору о новом сервере имен для своей зоны. Как правило, это делается через web-формы администрирования соответствующего держателя вышестоящего домена (domainpeople.com, nic.ru, etc).



Регистратор доменов RU-center

Очень часто забывают выполнить третий пункт, и тогда информация на вышестоящих серверах имен о NS'ах твоей зоны не совпадает с информацией о таковых, полученных непосредственно с них самих. Вся беда в том, что ответ твоих name-серверов считается авторитетным, тогда как ответ вышестоящих - нет. Это и называется lame delegation: dns-сервер является вторичным для зоны, но сам об этом не знает.



Сайт, посвященный настройке djbdns на разных платформах

Для работы dnscache должна быть создана необходимая иерархия каталогов при помощи утилиты dnscache-conf:

```
# dnscache-conf dnscache dnslog
/usr/local/etc/djbdns/dnscache 192.168.0.1
```

Аргументы, переданные утилите, очевидны: %26lt;учетная запись dnscache%26gt; %26lt;учетная запись dnslog%26gt; %26lt;рабочий каталог%26gt; %26lt;адрес, на котором будет слушать dnscache%26gt;

Можно указать dnscache работать в качестве форвардера пересылая все запросы к dns-серверу провайдера и кешируя результат:

```
# echo prov.dns.serv.ip %26gt;
/usr/local/etc/djbdns/dnscache/root/servers/@
# echo 1 %26gt; /etc/dnscache/env/FORWARDONLY
```

Разрешим нашей подсети доступ к кешу:

```
# touch
/usr/local/etc/djbdns/dnscache/root/ip/192.168.0
```

Для запуска dnscache жизненно необходимо наличие daemontools, подробнее о которых читай в статье про почтовый сервер в этом же номере.

```
# cd /usr/ports/sysutils/daemontools
# make install clean
# echo 'svscan_enable="YES"' %26gt;%26gt;
/etc/rc.conf
# mkdir /var/service
```

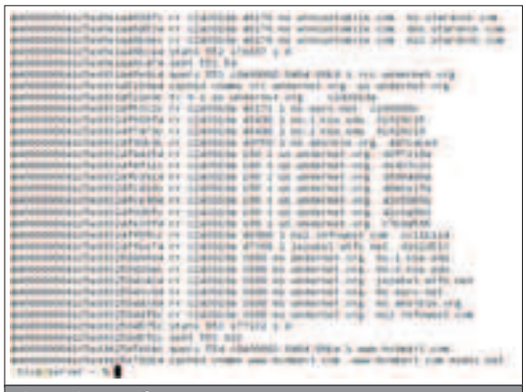
Запустим dnscache: создай символическую ссылку в каталоге, обслуживаемом установленными ранее daemontools:

```
# ln -s /usr/local/etc/djbdns/dnscache /var/service
```

Ждем секунд десять и передаем проверку работы dnscache утилите из daemontools svstat:

```
# svstat /var/service/dnscache
/var/service/dnscache: up (pid 13722) 13 seconds
```

»



dnscache в работе

Полезный ресурс по использованию djbdns

И все. Никаких проблем с named.conf. Софт djb вообще отличается тем, что его можно конфигурировать с помощью команды echo ;). Прописываем 192.168.0.1 в качестве nameserver:

```
# echo nameserver 192.168.0.1 %26gt; /etc/resolv.conf
```

DNS НА СТРАЖЕ ИМЕН

На очереди ответственный за зону DNS-сервер - tinydns, который конфигурируется похожим образом (заметь, что он вешается на внешний интерфейс):

```
# tinydns-conf tinydns dnslog
/usr/local/etc/djbdns/tinydns 11.22.33.44
# ln -s /usr/local/etc/djbdns/tinydns /var/service
# svstat /var/service/tinydns
/var/service/tinydns: up (pid 13821) 16 seconds
```

Как только речь заходит о настройке зоны DNS, в голове сразу возникают ассоциации с ворохом файлов в namedb: опечатки, когда ничего не работает из-за того, что забыл поставить точку, неочевидный синтаксис конфигурационных файлов и прочие радости сношения с пакетом BIND.

Можешь расслабиться. Разумеется, чтобы понять все тонкости настройки, тебе придется прочитать документацию к tinydns, однако смотри пример создания зоны с SOA двумя MX-записями и несколькими хостами. И если ты скажешь, что это сложно, то ты - мой злейший враг на всю жизнь ;).

```
# cd /usr/local/etc/djbdns/ext_tinydns/root
# vi data

.mydomain.ru:11.22.33.44:ns.mydomain.ru
.33.22.11.in-addr.arpa:11.22.33.44

@ank-pki.ru:11.22.33.44:mail.mydomain.ru:0
@ank-pki.ru:11.22.33.50:mx1.mydomain.ru:10

= host1.mydomain.ru:11.22.33.45
= host2.mydomain.ru:11.22.33.46
= host3.mydomain.ru:11.22.33.47

Cwww.mydomain.ru:host1.mydomain.ru
Cjabber.mydomain.ru:host2.mydomain.ru

# make
```

Для запуска dnscache жизненно необходимо наличие daemontools

Как видно, каждая запись предваряется типом. Точка - SOA, @ - MX, = - A и PTR запись одновременно, C - CNAME. Очевидно, мы присвоили хосту 11.22.33.44 имя ns.mydomain.ru, он же - mail.mydomain.ru. Второй почтовый сервер, mx1.mydomain.ru, расположен по адресу 11.22.33.50 в этом же домене. host1-host3 - хосты, на которых крутятся сетевые сервисы. Jabber и www - алиасы, чтобы было понятно, что на host1 работает Apache, а на host2 - Jabberd. После правки конфигурационного файла набери make, чтобы программа tinydns-data перечитала его и обновила бинарный файл data.cdb. tinydns осуществляет поиск в бинарном файле, а не текстовых конфигурационных файлов, благодаря чему работает намного быстрее того же named. Кстати, файл можно не править вручную: в том же каталоге присутствуют скрипты для добавления записей. И тут у тебя просто начинает рваться наружу вопрос: "Как же dnscache и tinydns будут уживаться на одной машине, если за резолвом машины из mydomain.ru dnscache полезет в интернет, а машина - вот она, тут же". "Склеить" два демона можно таким образом:

```
# echo 11.22.33.44 %26gt;
/usr/local/etc/djbdns/dnscache/root/servers/mydomain.ru
# echo 11.22.33.44 %26gt;
/usr/local/etc/djbdns/dnscache/root/servers/33.22.11.in-addr.arpa
```

Замечу, что если провайдер не делегировал тебе обратную зону, то заморачиваться с in-addr.arpa вряд ли имеет смысл, так как в глобальном масштабе твой сервер будет не прав: он необоснованно считает себя ответственным за обратную зону.

СИНХРОНИЗИРУЕМ ЗОНУ

Осталось только отдать нашу зону вторичному серверу. Как правило, чаще всего таким сервером бывает BIND. Конфигурируем axfrdns уже до боли знакомым образом:

```
# axfrdns-conf axfrdns dnslog
/usr/local/etc/djbdns/axfrdns
/usr/local/etc/djbdns/tinydns 11.22.33.44
```

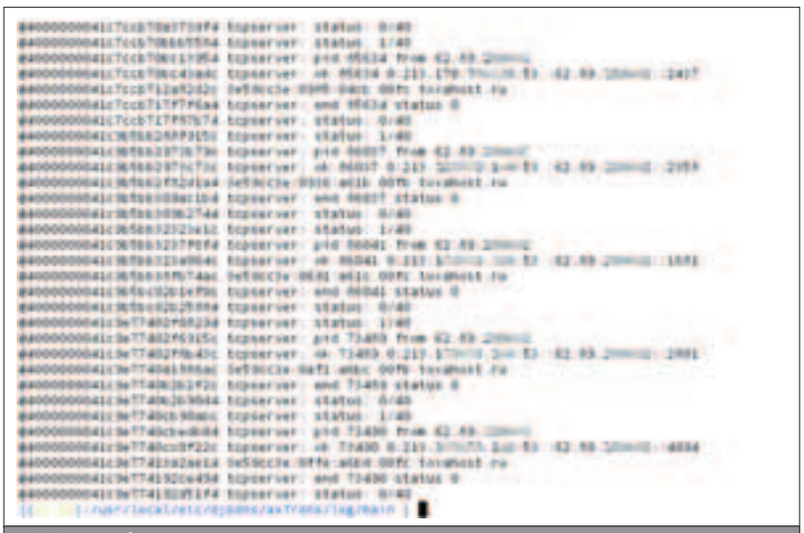
Добавился новый аргумент - путь к месту расположения конфигурационных файлов tinydns. Не пугайся, что axfrdns слушает на том же интерфейсе, что и tinydns: он bind'ит порт 53/tcp, тогда как tinydns - 53/udp (named для трансфера зоны использует протокол TCP). Настроив BIND как slave, разрешим ему стягивать зону.

```
# cd /usr/local/etc/djbdns/axfrdns
# echo '22.33.44.55:allow,AXFR="mydomain.ru"'
%26gt; tcp
# echo ":deny" %26gt;%26gt; tcp
# make
```

Вот так и была установлена переменная AXFR для хоста 22.33.44.55 (это адрес вторичного сервера) в значение "mydomain.ru". Это значит, что axfrdns позволит передачу соответствующей зоны машине 22.33.44.55.

```
# ln -s /usr/local/etc/djbdns/axfrdns /var/service
```

Ну а что если твой slave тоже использует tinydns? Например, если ты отвечаешь и за первичный, и за вторичный серверы. Тогда синхронизиро-



axfrdns в работе

вать зоны можно и без axfrdns. DJB предлагает не изобретать велосипед в виде 53/tcp, а использовать удобный и безопасный способ - rsync over ssh. Для этого потребуется всего три шага:

- Создаем на slave-сервере отдельного пользователя, который будет владеть базой имен (файл data.cdb), которая подлежит синхронизации. Ничто не мешает использовать рабочий логин:

```
# chown toxa
/usr/local/etc/djbdns/tinydns/root/data.cdb
# chown toxa /usr/local/etc/djbdns/tinydns/root
```

- На той же машине создадим заведомо некорректный файл data, чтобы после набора там make случайно не переписать синхронизированный data.cdb:

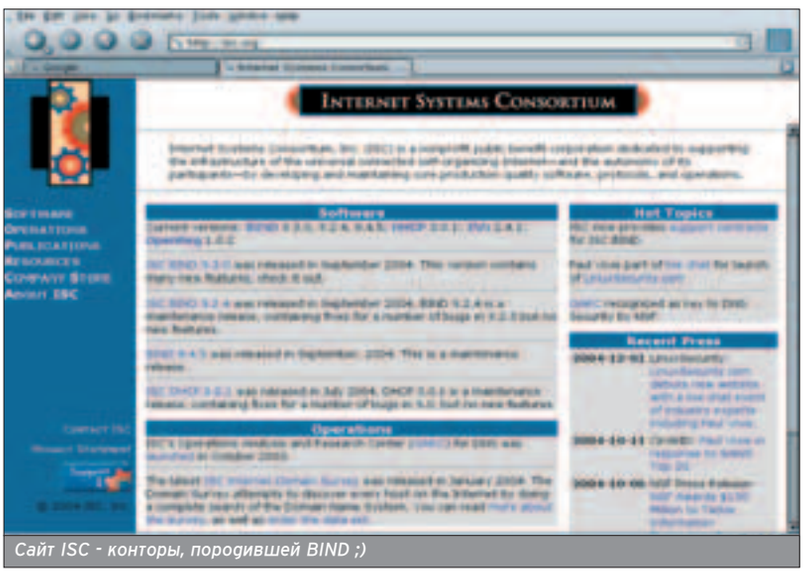
```
# echo thisserverisslave %26gt;
/usr/local/etc/djbdns/tinydns/root/data
```

- На первичном сервере добавляем в начало файла /usr/local/etc/djbdns/tinydns/root/Makefile строчки:

```
sync: data.cdb
rsync -az -e ssh data.cdb
toxa@22.33.44.55:/usr/local/etc/djbdns/tinydns/root/data.cdb
```

Теперь при внесении изменений на первичном сервере и при последующем обновлении data.cdb командой make данные обновятся и на вторичном сервере. Так как используется rsync (естественно, пакет rsync не входит в базовую поставку FreeBSD, поставь его из порта net/rsync), то по Сети будут переданы лишь изменения, а не весь data.cdb. Если ты настроишь ssh-авторизацию по ключам, у тебя даже пароля не попросят :).

Что получили в результате? Dnscache знай себе кеширует запросы из локальной сети, tinydns спокойно и размеренно рассказывает внешним машинам о хостах нашего домена, ну а axfrdns помогает тем несчастным, которые еще используют BIND, стягивать нашу name-зону. Теперь ты можешь смело оставить свой сервер жить своей жизнью - патчить djbdns тебе не придется. Ну а если возникнут какие проблемы, знай, что www.google.com не собирается менять свой URL ;).



Сайт ISC - конторы, породившей BIND ;)

Колисниченко Денис, dhsilabs@mail.ru

*NIX-УСКОРЕНИЕ

КАК СДЕЛАТЬ LINUX БЫСТРЕЕ?

Тебе надоело ждать, пока загрузится Linux на твоем стареньком компьютере, или просто хочется сделать его еще шустрее? В этой статье мы поговорим о способах повышения производительности твоей домашней Linux-системы. Для примера будем оптимизировать дистрибутив Mandrake Linux release 10.0 (Official) for i686 (ядро 2.6.3-7).

ПРИСТУПАЕМ К РАБОТЕ

Еще раз повторюсь: будем оптимизировать домашний компьютер, а не сервер Сети, поэтому все сказанное относится именно к домашнему компьютеру.

Для начала оптимизируем загрузку операционной системы - приятно осознавать, что теперь Linux загружается на десять секунд быстрее, не так ли? Для этого нужно сделать следующее:

- отключить поиск нового оборудования;
- отключить ненужные сервисы.

Займемся оптимизацией работы всей системы, а для этого:

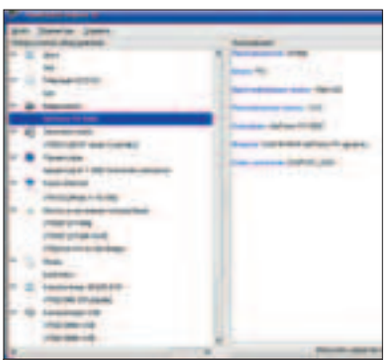
- по возможности использовать файловую систему ext2;
- "разогнать" винчестер;
- перекомпилировать ядро.

Теперь обо всем этом по порядку.

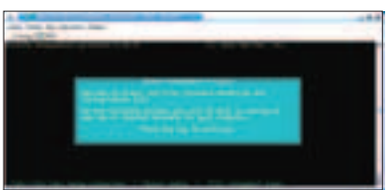
УСКОРЕНИЕ ЗАГРУЗКИ СИСТЕМЫ

Почему нужно отключать определение нового оборудования? Неужели ты каждый день устанавливаешь новую видеоплату или меняешь сетевую? Как правило, при сборке компьютера на него устанавливается операционная система, и о железе можно забыть до следующего upgrade'а. Даже если ты ярый любитель upgrade'а, думаю, тебе будет нетрудно два-три раза в месяц ввести команду Harddrake (в новых дистрибутивах - Harddrake2), чтобы Hard Drake обнаружил установленное устройство? Программа Harddrake в Linux Mandrake используется для поиска нового оборудования; в других дистрибутивах, в частности, Red Hat, используются другие программы, например, Kudzu. Что же касается целесообразности отключения Harddrake, то данная операция позволяет сэкономить от двух до пяти-семи секунд при загрузке системы в зависимости от конфигурации. Потом ее нужно будет запускать только после установки нового оборудования.

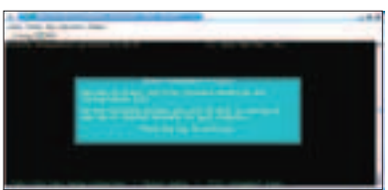
Harddrake отключается очень просто: заходишь в систему как пользователь root (или вводишь команду su в терминале), запускаешь программу Drakxservices и отключаешь Harddrake в списке сервисов. Если у тебя установлен Red Hat, для отключения ненужных сервисов используется конфигуратор redhat-config-services. При следующей загрузке Harddrake (или Kudzu) запускаться не будет.



Конфигуратор Drakxservices



Средство автоматического определения оборудования Harddrake2



Средство автоматического определения оборудования Kudzu

Теперь перейдем к самим сервисам. Их довольно много, и при первом знакомстве с окном Drakxservices часто хочется его закрыть сразу после торжественного первого открытия. А вот этого делать как раз и не нужно. Отключив неиспользуемые сервисы, мы убиваем сразу четырех зайцев.

- Ускоряем загрузку системы.
- Закрываем возможные дыры в системе безопасности, поскольку запуск сервиса пока имеет мало значения: его нужно правильно настроить. Часто бывает, что администратор, установив сервис, продолжает использовать его с параметрами по умолчанию. А этим и пользуются умные люди, которые учатся на ошибках других, а не на своих собственных.
- Ускоряем работу, поскольку ненужные сервисы уже не "отъедают" оперативную память и ресурсы процессора.
- Ускоряем останов системы, поскольку чем меньше сервисов запускается, тем меньше потом придется их останавливать.

Пройдемся по всем сервисам. Сервисы, отмеченные звездочкой, запускаются по умолчанию. Чтобы проще было ориентироваться, сразу обозначаешь минусом сервисы, которые нужно отключить, а те сервисы, которые можно отключить, но можно и не отключать, отмечаешь вопросительным знаком.

adsl

Управляет ADSL-соединениями. Тут все просто: если у тебя есть ADSL-соединение, тогда его нужно включить, в противном случае он даже не будет включен по умолчанию.

alsa (*)

Инициализирует расширенную звуковую архитектуру Linux (Advanced Linux Sound Architecture).

atd (*) (-)

Планировщик команд. Сейчас он тебе не нужен, поэтому смело отключай его. Когда он действительно понадобится, его активизация не займет много времени.

autofs (*) (?)

Средство автоматического монтирования сменных носителей (CD-ROM, дискета) по требованию. Желательно включить (точнее, не отключать). Хотя я предпочитаю монтировать CD-

ROM вручную, поэтому у меня этот сервис постоянно выключен - это уже кому как нравится. Определить потребность в нем очень просто: посчитай, сколько раз в день ты используешь CD-ROM. Если за день меняешь два-три диска, а в основном работаешь с сетью или интернетом, он тебе просто не нужен. Его вполне можно заменить сценариями cd-on (монтирование) и cd-off (демонтирование):

```
#!/bin/bash
mount -t iso9660 /dev/hdd /mnt/cdrom
```

В этом сценарии считается, что CD-ROM - это устройство /dev/hdd. Сценарий cd-off выглядит так:

```
#!/bin/bash
cd /
fuser /mnt/cdrom
umount /mnt/cdrom
eject
```

Конечно, это тривиальная версия такого сценария, но поскольку данная статья посвящена не монтированию дисков, а оптимизации всей системы, подробно останавливаться на этом сценарии не стану. Скажу только, что команда Fuser выводит того, кто в данный момент использует CD-ROM. Ведь пока какой-то процесс использует CD-ROM, ты не сможешь его размонтировать, а следовательно, и извлечь. Вторая команда выполняет размонтирование, а последняя - извлекает компакт-диск.

crond (*) (-)

Еще один планировщик, отключаем. Для домашнего использования он не нужен.

cups (*) (?), cups-lpd

Поддержка системы CUPS - Common Unix Print System, что в переводе означает "общая система печати UNIX". Если у тебя нет принтера, можешь отключить CUPS. А вот если принтер есть, то он без этого сервиса печатать не будет. Второй сервис добавляет в CUPS поддержку устаревшей системы печати lpd. Его нужно включить, если в сети есть гадкий утенок (компьютер с Unix'ом, на котором установлена lpd), к тому же этот утенок хочет печатать на твоем CUPS-принтере.

devfsd (*)

Это своеобразный уборщик твоей системы: удаляет мусор, его не отключай.

dm (*)

Менеджер дисплея (Display Manager); если ты планируешь использовать графический интерфейс (X Window), не стоит отключать данный сервис.

fam (*)

Демон следит за изменившимися файлами, используется GNOME и KDE, также отключать нежелательно.

harddrake (*) (-)

Об этом мы уже говорили... Отключаем.

httpd (*) (-)

Это web-сервер Apache. Зачем тебе на домашней машине web-сервер??! Если ты программируешь для web, можно запускать его уже после загрузки системы (Service httpd start), чтобы он не занимал ресурсы системы. Пока он не нужен. Вердикт: отключить.

internet (*) (-)

Устанавливает соединение с провайдером при загрузке системы и обрывает его при завершении работы. Когда нам нужно, мы и сами в состоянии дозвониться до провайдера.

Отключаем.

iptables (*) (-)

Это файрвол (firewall). Пока мы его отключим. Я не говорю, что iptables - это плохо, но пока ты его не настроил, лучше его отключить.

jserver (*) (-)

Вот этот сервис меня насмешил: несколько раз я запускал Drakxservices, но его не трогал: думал, что он связан с Java, а оказалось - с ... японцами. Первая буква от слова Japanese. Если ты не японец, смело отключай этот ненужный сервис.

keytable (*)

Этот загружает выбранную раскладку клавиатуры, указанную в файле /etc/sysconfig/keyboard. Нужен для нормальной работы большинства систем.

kheader (*) (?)

Автоматическая регенерация заголовков ядра. В принципе, он и не сильно нужен, но я его оставил.

Не стоит отключать проверку ext2-разделов программой Fsck.

DANGER!

■ Внимание! Использовать Hdparm нужно очень аккуратно, поскольку в неумелых руках он может стать причиной потери данных. Лучше экспериментировать с Hdparm сразу после установки системы, когда у тебя еще не накопились важные данные.

mailman (*) (-)

Я бы назвал его SpamMan. Легализированный пособник спаммерам. В общем, средство-менеджер для списка рассылок. Отключаем.

mtink (?)

Низкоуровневый драйвер для принтера Epson. Если у тебя Epson, стоит включить этот сервис.

mysql (*) (-)

Сервер баз данных MySQL. На домашнем компьютере он не нужен.

netfs (*) (-)

Обеспечивает монтирование удаленных сетевых файловых систем (NFS, SMB, NCP (NetWare)). Может пригодиться для работы в локальной сети, но если ты подключен только к интернету, он не нужен.

netplugd

Демон netplugd обрабатывает различные события соединений, получаемые от ядра Linux. Например, потеря соединения или получение сигнала несущей.

network (*)

Поддержка Сети. Отключать нельзя, поскольку в Unix, как и в Linux, даже функции печати (не говоря уже об X Window) требуют поддержки Сети.

nfs (*) (-), nfslock (*) (-)

Реализуют поддержку NFS (Network File System). Отключаем оба.

numlock (*)

Безобидный сервис, включающий режим ввода цифр на дополнительной клавиатуре.

oki4daemon

Если у тебя Windows-принтер OKI, включай его.

partmon (*) (?)

Лучше не отключать: он проверяет, скоро ли будет заполнен раздел. Хотя... вводи почаще df -h и будешь получать более полезную информацию об использовании разделов.

postfix (*) (-)

Агент доставки почты. На домашнем компьютере не нужен.

proftpd (*) (-)

Файловый сервер ProFTD. Зачем он тебе на домашнем компьютере???

random (*)

Улучшает качество генерации случайных чисел. Включи его: запуск не займет много времени.

rawdevices (*) (-)

Назначает raw-устройствам block-устройства. Нужен для Oracle и некоторых DVD-проигрывателей.

smb (*) (-)

Если не планируешь подключаться к сети Microsoft, отключи его. Даже если планируешь, все равно отключи: включишь, когда настроишь Samba. Надеюсь, скоро мы поговорим и о настройке Samba, но об этом - в следующей статье.

sshd (*) (-)

На домашнем компьютере SSH (Secure Shell) просто не нужен. А если хочешь уберечь свои данные от родственников, придумай пароль поинтересней, чем 123456 или qwerty. »

syslog (*)
Это системный журнал. Не нужно отключать его!

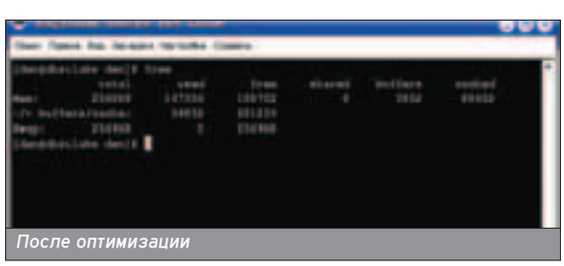
xfs (*)
X Font Server - сервер шрифтов X Window. Не отключать!

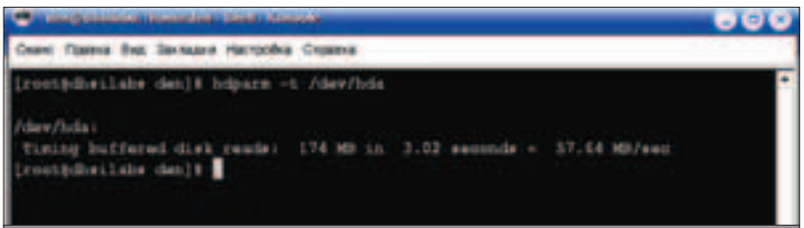
xinetd (*)
Суперсервер xinetd - это основа основ, так как без него не будет работать большинство сетевых сервисов, таких как POP3, IMAP, FTP (если он не запускается отдельно) и др. Почему xinetd называется суперсервером? Да потому что он отвечает за установление TCP-соединения, то есть прослушивает пакеты и запускает необходимые программы для обработки информации. Таким образом, получается, что сервер inetd (xinetd) управляет другими серверами и потому называется суперсервером. Например, если в запросе клиента будет требование установить соединение с двадцать первым портом, то суперсервер вызовет сервер ftp, конечно, при условии, что соединение с 21-м портом разрешено (в противном случае клиент получит сообщение Connection refused). Конечно, не все так просто, как я описал, но моя статья посвящена оптимизации Linux, а не серверу xinetd, поэтому подробно останавливаться на нем не буду.

В зависимости от установленных пакетов сервисы могут отличаться. Например, может быть установлен сервер DNS (сервис Named), вместо Postfix может использоваться Exim или Qmail, а вместо Proftpd и Vsftpd или Pure-ftpdи т.д.

Результат оптимизации: отключено минимум 17 сервисов! Попробуй перезагрузить компьютер. Ну как, быстрее? Если говорить точнее, то загрузка Linux на моей машине (Duron 1,6Mhz 256MB/40 GB Maxtor) стала на 8 секунд быстрее. Если до оптимизации загрузка с момента запуска сервисов (с момента появления надписи "Нажмите I для интерактивной загрузки") до появления графического менеджера входа в систему занимала 17 секунд, то после оптимизации - всего 9.


До оптимизации


После оптимизации


Скорость чтения информации

Результат оптимизации: отключено 17 сервисов!

Теперь о памяти. Система загружается, я вхожу в систему, запускаю терминал и ввожу команду Free. До оптимизации у меня свободными были 52 Мб оперативной памяти, а после отключения ненужных сервисов - 108 Мб. 8 секунд и 56 Мб свободной памяти - вот результат оптимизации

Во время запуска системы производится автоматическое монтирование файловых систем, указанных в файле /etc/fstab. Монтирование файловой системы занимает определенное время - обычно не очень много, но если файловая система не одна, можно выиграть еще немного времени. Для этого открой файл /etc/fstab и закомментируй строки, описывающие файловые системы, которые ты редко используешь (или вообще не используешь). Отключать проверку ext2-разделов программой Fsck я не рекомендую - те пару секунд не стоят возможной потери данных.

ФОРСИРОВАНИЕ ВИНЧЕСТЕРА

■ Тут ситуация двойственная. Форсаж подразумевает работу какого-либо устройства или механизма на пределе, из-за чего механизм хоть и работает быстрее, но изнашивается с еще большей скоростью. Не будем издеваться над винчестером и тем самым продлим срок его жизни. Просто старые дистрибутивы (и некоторые новые) не включают определенные функции, например, DMA или Multcount. Большинство новых дистрибутивов по умолчанию используют оптимальные параметры для твоего винчестера. Конечно, можно заставить его работать еще быстрее, но тогда он может работать нестабильно, периодически будут всплывать ошибки чтения или записи. Поэтому займись установкой оптимальных параметров, если этого не сделал дистрибутив. Сначала узнаем скорость работы винчестера:

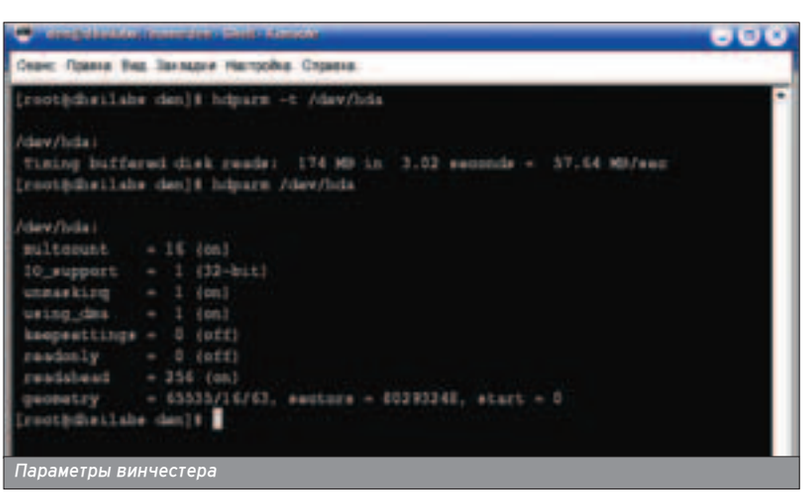
```
# hdparm - t /dev/had
```

Эту команду нужно вводить от имени пользователя root. Я, например, получил результат 57,64 Мб/с. Такой результат меня устраивает, поэтому я даже не пытался его увеличить, а просто еще раз запустил Hdparm, чтобы просмотреть параметры винчестера.

```
$ hdparm /dev/hda
```

У меня все нормально:
- передача нескольких секторов (multcount) за такт включена (16 секторов);
- включена поддержка 32-битного ввода/вывода;
- включено использование DMA.

На старом дистрибутиве (правда, для другого винчестера) я получил следующее:


Параметры винчестера

```
/dev/hda:

multcount = 0 (off)
I/O support = 0 (default 16-bit)
unmaskirq = 0 (off)
using_dma = 0 (off)
keepsettings = 0 (off)
nowerr = 0 (off)
readonly = 0 (off)
readahead = 8 (on)
```

Тут полный бардак: Multcount выключен, DMA не используется, поддержка 16-битного ввода/вывода тоже выключена. С такими параметрами винчестер выдавал лишь 3,75 Мб/с. Исправить это помогла команда

```
# hdparm -d1m8c3u1 /dev/hda1
```

Теперь разберемся, что же сделала эта команда. Во-первых, мы включили DMA (d1), потом разрешили передавать больше одного сектора за такт (8) и включили 32-битный доступ к диску (команда c3). Кстати, параметр u1 полезен и в тех случаях, когда у тебя начинает "заикаться" xmms во время прослушивания музыки. Можно поэкспериментировать и с другими параметрами Hdparm, узнать о которых можно в справке (man hdparm). Для сохранения параметров контроллера IDE используется команда

```
# hdparm -k 1 /dev/hda
```

При перезагрузке системы параметры IDE теряются, поэтому команду "разгона" винчестера нужно поместить в сценарий запуска системы. Просто добавь команду вызова Hdparm в файл /etc/rc.d/rc.local. Этот способ является универсальным, поскольку он позволяет установить отдельные параметры для разных жестких дисков, если их несколько. Второй, менее универсальный способ заключается в редактировании файла /etc/sysconfig/harddisks, в котором можно задать общие параметры для всех жестких дисков. Есть еще один подводный камень: при пробуждении системы в нормальное состояние после "сна" параметры контроллера также сбрасываются. Этого можно избежать, если подправить файл конфигурации демона Apmd, который отвечает за управление питанием. Параметры контроллера IDE, которые устанавливаются при переходе системы в "спящий" режим и при выходе из него, задаются строками HDPARM_AT_SUSPEND и HDPARM_AT_RESUME в файле конфигурации /etc/sysconfig/apmd. Файлы конфигурации, расположенные в каталоге /etc/sysconfig, имеются только в системах, подобных Red Hat -- это Red Hat Linux, Mandrake Linux, SuSE Linux, ASP Linux, Back Cat Linux, ABI Linux и другие.

ПЕРЕКОМПИЛИРОВАНИЕ ЯДРА

Подробно рассматривать процесс перекомпилирования ядра не будем - этому посвящено очень много статей. В принципе, ядро можно и не перекомпилировать, но все-таки лучше это сделать. Разработчики дистрибутива не знают, на какой компьютер будет устанавливаться их дистрибутив, поэтому ядра идут универсальные - для процессора 586/686. У тебя же установлен совершенно другой процессор. Поэтому первое, что нужно сделать, - это в утилите конфигурирования ядра установить свой тип процессора. Затем пройтись по всем функциям ядра и решить, нужны ли они тебе. Каждая функция "отъедает" кусочек оперативной памяти. Если функция не нужна (или не нужна в ближайшее время), ее можно выключить или, по крайней мере, попытаться включить в состав ядра в виде модуля. В этом случае она не будет занимать память, когда она не нужна, а будет загружаться только по требованию ядра. Наоборот, те функции, которые тебе точно необходимы, нужно стараться включить в ядро (не в виде модуля!). В этом случае они будут работать быстрее. Только тут важно не перестараться, а то можно получить гигантское и неповоротливое ядро.

НЕ ХВАТАЕТ ПАМЯТИ?

Система может изрядно притормаживать, если ей не хватает памяти. Возможно, у тебя всего лишь 128 Мб оперативки, а при создании swap-раздела ты пожадничал и отвел для него всего 64 Мб (или вообще не создавал его). Что ж теперь делать? Неужели опять переразбивать винчестер? Можно просто создать swap-файл. Для этого сначала создай пустой файл /swap/sw-file (в примере - размер 32 Мб) с помощью команды dd:

```
dd if=/dev/zero of=/swap/sw-file bs=1k count=32768
```

Стоит отметить, что эта команда читает данные с устройства /dev/zero и записывает их в файл /swap/sw-file. В качестве данных будет просто поток нулей, причем не чисел ноль (ANSII-код 48), а неотображаемых символов NULL (ANSII-код 0). Данные читаются и записываются блоками по 1 Кб (bs=1k), и общее количество блоков равно 32768. Таким образом на выходе будет получен файл размером 32 Мб, заполненный символами NULL. Действия по созданию такого файла очень сходны с действиями, производимыми программой Fdisk при создании нового раздела. После этого отформатируй данный файл под swap:

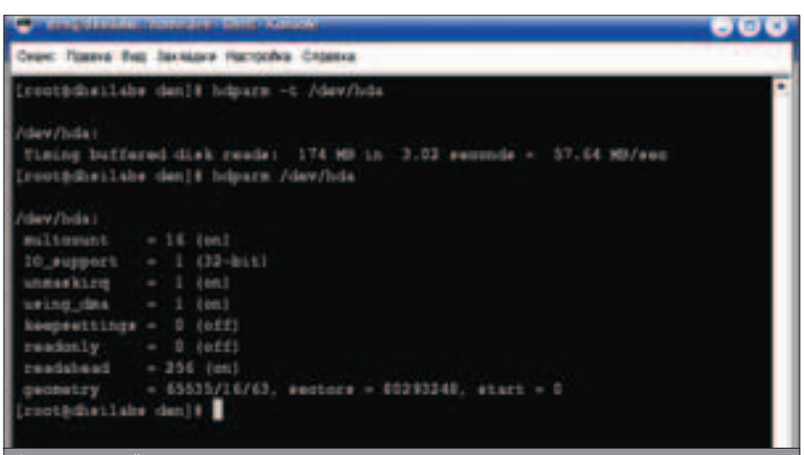
```
mkswap /swap/sw-file 32768
```

После создания файла подкачки его нужно активизировать. Команда Swapon -a включает все разделы свопинга (описанные в файле /etc/fstab), а команда Swapon <раздел> включает только конкретный раздел. Команда Swapon -a обычно помещается в сценарий загрузки системы. Обычно это /etc/rc.d/rc.sysinit для систем, использующих инициализацию типа SysV -- RedHat, Mandrake, Debian (хотя RedHat и Mandrake используют несколько модифицированную схему инициализации, но суть та же) или /etc/rc/rc.S для BSD-подобных Linux-систем (Slackware). Для подключения нашего файла подкачки необходимо выполнить команду

```
swapon /swap/sw-file.
```

На этом оптимизацияю Linux прошу считать законченной. Наслаждайся скоростью!

После создания файла подкачки его нужно активизировать.



Создание файла подкачки

Content:

DESKTOP

Петр "Roxton" Семилетов (tea@list.ru)

LINUX НА ДЕСКТОПЕ

РАЗБИРАЕМСЯ В ПРИЕМЛЕМОСТИ LINUX ДЛЯ РАБОЧИХ СТОЛОВ СРЕДНЕСТАТИСТИЧЕСКИХ ГРАЖДАН

Периодически в западной прессе появляются статьи на тему того, готов Linux для десктопов или не готов. Этот материал призван взвесить все "за" и "против" и дать объективное заключение по этому интересному вопросу.

BSD НЕ БУДЕТ

О различных вариантах BSD - OpenBSD, FreeBSD, NetBSD вопрос даже не ставят, что, надо признать, справедливо. Сложно представить себе рядового пользователя, который установил бы себе на компьютер FreeBSD, что в наши дни возможно скорее всего только как результат хорошей работы стильного чертенка (редкий интернет-магазин, посвященный Open Source, не продает сейчас черные футболки с этим логотипом BSD).

BSD следует рассматривать как чисто профессиональную, узко специализированную систему, пусть даже в ней и присутствуют элементы, ориентированные на массового пользователя. Она востребована именно среди профессионалов и выполняет соответствующие функции - например, работает на серверах.

Другое дело – более универсальный Linux. Разумеется, BSD тоже универсальна, однако база приложений, ориентированных на нужды пользователей, у Linux несомненно больше, да и ядро поддерживает больше железа, популярного среди массового пользователя.

В этом году количество Linux-систем в десктоп-секторе превысило количество систем Macintosh. А западные аналитики продолжают спрашивать, готов Linux для рабочего стола или не готов? За исследования на эту тему выдают большие и солидные обзоры (впрочем, такие можно встретить и в отечественной прессе). Журналист, впервые услышавший о Linux только вчера, берет кучу дисков с дистрибутивами и усаживается за тот злополучный "тестовый" компьютер, который есть в каждой редакции. Этот подобный чудовищу доктора Франкенштейна аппарат, он же невообразимое сочетание "левых" и ненужных железок, гордо именуется в статье не иначе как "наша тестовая машина". На нее-то журналист и устанавливает дистрибутивы один за другим.

Какое мнение при этом у него может сложиться - трудно сказать, потому что в статье такие "исследователи" ограничиваются общими фразами, указывают на некую "сырость", жалуются на мелочи совместимости с форматами MS Office, а под конец повергают читателя в шок сообщением о том, что в Linux не запускаются их любимые игры!

НЕ ВСЕ КОТУ МАСЛЕНИЦА

Другая сторона монеты - это статьи оптимистически настроенных товарищей линуксоидов, которые рисуют идиллические картины и призывают устанавливать Linux, но совсем не упоминают подводные камни, которые, увы, пока существуют. В первую очередь это касается железа.

Однако под Linux в эмуляторе можно запускать даже старые DOS-игры вроде Alone In Dark

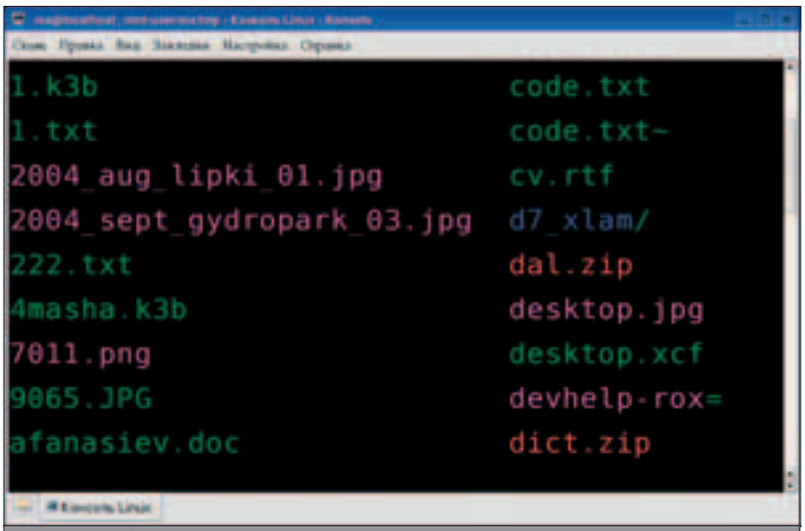
Опытные линуксоиды подбирают себе железо исходя из его совместимости с Linux. Самое новое железо, которое может еще не поддерживаться ядром, они отвергают. USB-модемы и софт-модемы тоже оказываются под запретом, потому что мало кого прельщает перспектива провести выходные дни в трудах, сравнимых с чтением египетских иероглифов до обнаружения Розеттского камня.

Вот пример из жизни. Я поставил себе новое ядро - старый LPT-сканнер перестал работать, потому что чудесным образом поменялся адрес LPT-порта. Было 0x378, стало 0x278. Хорошо, что я знаю, где править это значение - в etc/sane.d/mustek_pp.conf, а что бы делал среднестатистический десктоп-пользователь? Стал бы названивать знакомому гуру.

Новое железо (кроме видеокарт и модемов) в Linux начинает работать полноценно только тогда, когда в ядре появляется его поддержка. Нет поддержки - может работать, а может не работать. А может работать, но не так, как ожидается. Если разработчики железа обычно прилагают к своему продукту CD с драйверами, то модули для ядра Linux на таком диске ты вряд ли обнаружишь (хотя бывают приятные исключения). Значит, придется ждать свежих версий ядра.

Итак, существуют определенные трудности с поддержкой железа. А люди, которые собирают компьютеры на продажу, о совместимости с Linux задумываются очень редко. Если ты пойдешь в отечественный компьютерный магазин и попросишь собрать тебе компьютер на заказ, при этом целиком положишься на вкус менеджера (они там обычно все менеджеры :)), то получишь компьютер, который для полноценной работы с Linux не очень подходит. Менеджеры обычно знают одну марку процессора - это "целерон", и одну модель модема - просто модем, под которым подразумевается, разумеется, внутренний софт-модем. Та самая дешевая трескучая штучка, которая при конфликте DMA-каналов может творить чудеса и сводить тебя с ума.

Покупаешь такой компьютер, приносишь домой, устанавливаешь на него свежий Linux и пытаешься выйти в Сеть. Не тут-то было! Свежий Linux своими силами не поддержит твой модем, однако из Сети ты сможешь скачать чудо-драйверы, которые непременно помогут. Ты идешь к другу, и, не замечая его некоторой злобы, качаешь по его dialup нужные драйверы. Эти драйверы бесплатные, но с ограничением скорости, а если хочешь оторваться на всю катушку - плати деньги и получай полноценные драйверы. Надо сказать, что не разработчики Linux придумали это.



Хотя может быть и такое - только если сам захочешь

Наморочавшись с софт-модемом, ты вдруг обнаруживаешь, что программисты из ATI не успели написать драйвер поддержки 3D к твоему новому Radeon, и ты долгое время, пока дрова не обновятся, наслаждаешься качественной 2D-картинкой, иногда ради интереса запуская трехмерные игры, чтобы посмотреть, как они работают в программной эмуляции OpenGL.

Между прочим, среди моих знакомых линуксоидов в последнее время наблюдается тенденция переходить на видеокарты от nVidia. А в недавнем интервью на linuxquestions.org разработчики из nVidia сообщили, что Linux используется примерно на 15-20% рабочих станций внутри компании. Кроме того, компания владеет крупнейшим центром по симуляции чипов, построенным на основе Linux. Когда вышел третий DOOM, под Linux в него можно было играть только на видеокартах от nVidia - драйверы от ATI не позволяли этого, пока не была выпущена их новая версия.

У меня самого - Radeon 8500, в нем и 2D, и 3D работают, хотя 3D-часть драйверов не сравнить с ее аналогом под Windows. Это реальное положение вещей - фанатики могут забросать меня гневными письмами, но я не смотрю на мир сквозь розовые очки.

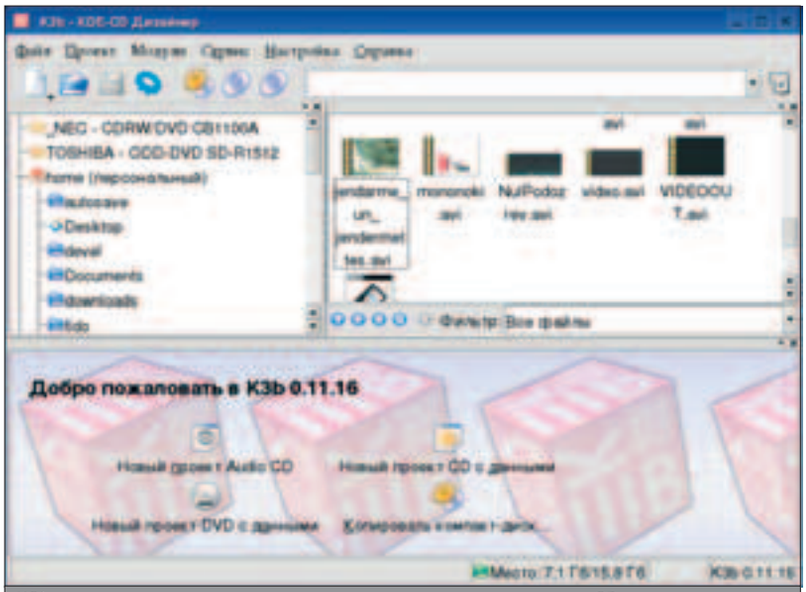
Я работаю в Linux с весны 2001 года, и только год назад Linux стал моей системой по умолчанию. Человек выбирает ту систему, в которой ему работать приятнее. На сегодня для меня оптимальным выбором является именно Linux, потому что при хорошо подобранном железе это действительно очень удобная и полезная система. Вопрос в том, хочет ли пользователь - рядовой пользователь, пользователь "десктопа" - подбирать это самое железо? Однако за него могут подобрать другие, а именно - производители компьютеров. Процесс уже пошел: все больше и больше компаний, продающих компьютеры в готовой сборке, объявляют о своей поддержке Linux.

Именно этот фактор важнее всего в продвижении Linux на десктоп. Кто такие в большинстве своем пользователи "десктопов"? Это чайники. Я не употребляю это слово в негативном смысле. Просто в мире живет огромное количество людей, которые являются потребителями компьютерной техники и которые относятся к ней как к бытовым приборам вроде музыкальных центров или телевизоров.

Когда у тебя ломается телевизор, ты везешь его в сервисный центр или вызываешь на дом телемастера. В особо тяжелых случаях - покупаешь новый. Так и у чайников, для которых компьютер - это прибор, который делает то, что и должен делать, но не более того. Чайник не станет разбирать компьютер, чайник не станет устанавливать систему. Чайник может быть специалистом в какой-нибудь одной области - например, в совершенстве владеть дуэтом Adobe Photoshop и Quark XPress, однако при этом он не в силах отформатировать винчестер. Чайник-фидошник с закрытыми глазами настроит чудесную связку Fastecho, TMail (непременно NT)

■ Рассуждать о том, готов Linux для декстопа или не готов, можно только учитывая специфику конкретного дистрибутива. Возьмем собираемый из исходников Gentoo Linux - это дистрибутив для энтузиастов или специалистов. Трудно представить себе ситуацию, в которой он стал бы массовым продуктом и широко распространенной именно на "рядовых" десктопах системой. А вот ALT Linux, SUSE, Fedora Core или Mandrake Linux - другое дело. Это совсем не значит, что одни дистрибутивы лучше, а другие хуже. Каждому свое.

»



K3b - с ее помощью линуксоиды записали уже не один миллион CD

и GoldEd, но вопросом о том, как записать что-то на болванку, будет осаждать не одну эхо-конференцию.

ИДЕАЛЬНЫЙ КОНЕЧНЫЙ ПОТРЕБИТЕЛЬ

■ Существует миф о сложности установки Linux. Это миф! Любая система по-своему сложна в установке. Сложна для того, кто не умеет ее устанавливать - для чайника. Поэтому с инсталляцией Linux проблем может быть столько же, сколько при установке любой другой системы. Обычно системы устанавливаются только теми пользователями, уровень компьютерных знаний которых равен "продвинутому пользователю" или чуть выше. В этом случае установить Linux не сложнее, чем установить Windows.

Итак, чайник является идеальным конечным потребителем, у которого есть определенные нужды. Во-первых, чайнику нужна простота взаимодействия с компьютером. Под этим подразумевается графический интерфейс. Две наиболее популярные в Linux графические среды - KDE и Gnome. У кого там слюнки потекли при упоминании этих названий? Конечно же, у пользователей других систем. Кажется, распространенная ассоциация Linux с черным экраном, на котором сообщения выводятся ярко-зелеными буквами, канула в лету.

Не все так просто. Когда-то в США провели такое исследование. Группа чайников познавала Linux. Эти самые добровольные респонденты пришли в крайнее недоумение потому, что для обозначения файловых каталогов вместо слова "folder" (как в Windows) в Linux используется слово "directory". Я думаю, что таким пользователям надо давать всего одну кнопку на экран. Тогда они точно не запутаются.

Если не впадать в крайности, то можно с уверенностью сказать, что графические интерфейсы Linux находятся на очень высоком уровне и обеспечивают пользователей всем необходимым, делая работу удобной и стабильной.

НЕДОВОЛЬСТВО ОТ ПРЕСЫЩЕНИЯ?

■ Однако находятся люди, которые и тут найдут проблему. Проблему выбора "десктопа" - Gnome и KDE? Якобы неким будущим (зачем будущим, когда УЖЕ есть разработчики) разработчикам так будет проще - писать приложения только для Gnome или только для KDE. А KDE-программы нормально работают и в Gnome, и наоборот. На мой взгляд, эта "проблема" не имеет достаточных оснований под собой. Кому удобно использовать строгий и эстетичный Gnome - те будут работать в нем. Любители же разных "наворотов" и невероятного количества опций для настройки всего по своему усмотрению предпочтут KDE. Возможность выбора - одно из преимуществ Linux. И не надо пытаться выдать это преимущество за недостаток.

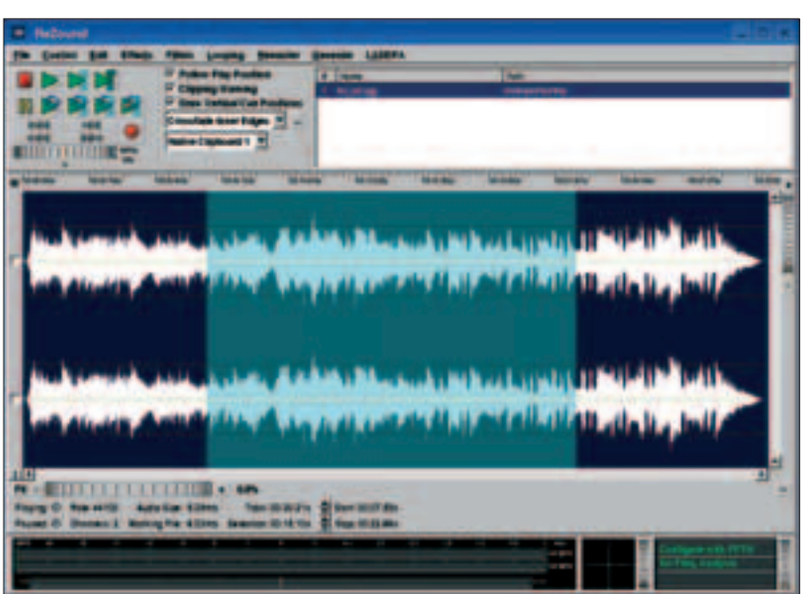
Современный графический интерфейс десктопа Linux - это RSS-новости в панели, это сводки погоды прямо на рабочем столе, вывод данных от системных датчиков температур, анимация в качестве обоев, векторные иконки (каждая из которых может занимать хоть весь экран), прорисовка с 3D-акселерацией и визуальные эффекты, прямо как в компьютерах из фантастических фильмов. Какие претензии могут быть к KDE и Gnome у злопыхателей и очернителей - не знаю. Нужен простой рабочий стол? Получи простой. Нужен сложный - получи сложный, с наворотами. Что захочешь, то и будет. Сам себе хозяин. Рабочая среда - это еще не все, нужно программное обеспечение для простых смертных, а не одни только компиляторы да отладчики? Отвечаю.

КАЖДОМУ - ПО ПОТРЕБНОСТЯМ

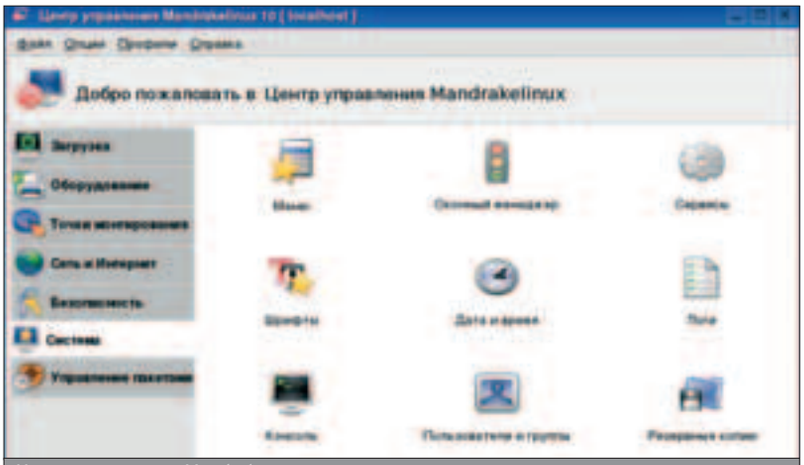
■ В Linux практически для любой программы, которая обитает в мире конкурирующей платформы, существуют аналоги. Суди сам. Запись CD: под Windows - Nero. Под Linux - cdrecord и графические оболочки к нему (например, K3b, Eroaster).

Панельный и многофункциональный файловый менеджер: Windows - Total Commander. Linux - Krusader, Konqueror. Консольный файловый менеджер: Windows – FAR, Linux - Midnight Commander. Windows - почтовые клиенты Outlook и The Bat!. Linux - Evolution, MKMail, Sylpheed. Видеоплейер: Windows - выбор продвинутого пользователя BSPlayer, Linux - плейеры с аналогичными и превосходящими функциями Mplayer, Xine. Список соответствий можно продолжать до бесконечности.

Но бывают и исключения. Исключения, относящиеся непосредственно не к десктоп-нише, а к нише профес-



Звуковой редактор ReZound



Центр управления Mandrake

сиональной. Отечественная бухгалтерия - 1С. Под Linux нет 1С. Профессиональная звукорежиссура для Windows и Mac предлагается по крайней мере четыре продукта, аналогов которым в мире Linux нет. Я говорю о Steinberg Cubase, Steinberg Nuendo, Sonar/Cakewalk и Samplitude. Аналог Sony SoundForge или WaveLab в Linux все-таки есть - это ReZound.

Профессиональные программные продукты никогда не были массовыми. Они сложны, стоят на порядок дороже, чем обычный софт, и нельзя принимать во внимание их наличие или отсутствие при анализе готовности к использованию на десктопе операционной системы и сопутствующего ей ПО.

В целом, несмотря на некоторое частичное отставание в поддержке нового железа (хотя версии Linux для 64-битных процессоров AMD появились раньше других массовых систем), Linux является системой, пригодной для полноценного использования на десктопах. Если бы это не было так, то крупные компании, такие как Novell, RedHat, Mandrake, не видели бы спроса на десктоп-ориентированные дистрибутивы и не выпускали бы их. Однако факты убеждают нас в обратном. Выпускаются как десктоп-ориентированные дистрибутивы, так и те, которые заточены под использование на серверах или в бизнесе. К "десктопу" начинает склоняться даже Debian, которая работает над приближением своей установочной утилиты к конечному пользователю - к чайнику, а не к гуру.

Дистрибутивы уже давно оснащаются центрами управления настроек системы. В RedHat/Fedora Core это Anaconda, в Mandrake - Mandrake Control Center, в SUSE - YAST. Рабочие среды KDE и GNOME тоже начали интегрировать в себя функции по конфигурации Linux - не только ядра и его модулей, но и подсистемы видео.

А ИМ ЛИШЬ БЫ ПОИГРАТЬ

Итак, хватает и средств настройки, и программного обеспечения, однако нельзя не учитывать еще один фактор - игры с большими бюджетами. "Десктоп"-сектор прочно ассоциируется у нас не только с офисными пакетами или видеоплейерами, но и с коммерческими играми от крупных издателей. Windows - не лидирующая платформа для производителей коммерческих игр. Ниша игровых консолей - вот основной рынок сбыта крупных проектов. И только после нее - Windows. Под Linux на коммерческих началах выпускаются в основном те игры, графика которых построена на OpenGl, то есть чаще всего на движках от idSoftware.

TuxRacer. А под Linux бывают и такие некоммерческие игры

Выпуск коммерческих игр под Linux зависит от двух факторов. Фактор первый - спрос. Спрос уже есть - Linux сейчас вышел в широкие массы, и многие используют его в качестве игровой платформы Windows, которую держат на машине параллельно с Linux, или пытаются запускать Windows-игры через эмуляторы вроде WINE. Фактор второй, сдерживающий - драйверы видеокарт, вернее их 3D-часть. Пока производители видеокарт не станут поддерживать Linux-драйверы на том же уровне, что и под Windows, разработчики игр будут считать область создания игры под Linux проблемной.

Нередко бывает и наоборот: разработчики игр вынуждают производителей видеокарт обеспечивать более полную поддержку своих изделий под Linux. Это хорошо видно даже на примере выпуска DOOM 3. Поскольку с последними к тому времени драйверами от ATI новый DOOM не работал, то команде разработчиков драйверов в ATI пришлось поторопиться, чтобы выпустить свежие драйверы, с которыми DOOM заработал нормально.

Жил одни знаменитый барон, который вытащил сам себя из болота за волосы. Так и со многим в компьютерном мире. Чем больше разработчиков коммерческих игр будут втягиваться в разработку многоплатформенных игр (я не говорю о разработке игр именно под Linux), тем более ударно будет развиваться эта отрасль и тем больше она будет влиять на распространение Linux в десктоп-сегменте рынка.

"Играбельность" системы ценится в основном молодежью, которая рассматривает компьютер в первую очередь как средство развлечения, а не работы. А такой молодежи подавляющее большинство, именно поэтому и слышны иногда мнения, что "Раз в Linux не работает моя любимая игра, Linux мне не нужен". Те же, кто используют компьютер именно для работы, могут оценить Linux на десктопе по достоинству.

Пока же Linux-десктопы медленно, но уверенно завоевывают рынок по крайней мере за пределами нашей страны. Например, на Walmart'е продается ряд дешевых компьютеров на базе Linux, в частности на Sun's Java desktop и Xandros Desktop Operating System. Подобные Linux-решения начинают появляться и у нас - смотри пресс-релизы на altlinux.ru и www.asplinux.ru. Однако самым авторитетным мнением о годности Linux для десктопов можно считать мнение муниципалитета Пекина, который переходит на Red Flag Linux.

Колисниченко Денис, dhsilabs@mail.ru

ВЕЧНАЯ ДРУЖБА

WINDOWS И LINUX НА ОДНОМ КОМПЬЮТЕРЕ

Скорее всего, Linux – это не единственная операционная система на твоем компьютере. Как минимум, их две – Linux и Windows. Давай заставим их подружиться на одном компьютере.

ЗАСТАВИМ ИХ ЖИТЬ ВМЕСТЕ!

Как установить операционные системы, чтобы они благополучно сосуществовали? Начнем с создания разделов на твоем винчестере. Идеальная схема:

* Первичный, активный, FAT32 – для Windows. Сюда можно установить любую версию Windows – от Windows 98 до Windows XP. Размер этого раздела не должен быть очень большим: на нем будет только операционная система. Минимальный размер – 1,3 Гб (нужно для установки Windows XP), максимальный – 3 Гб.

* Первичный, Linux swap – только не надо размещать раздел подкачки физически "в конце" диска, лучше ближе к началу – так работа с виртуальной памятью будет быстрее. Размер этого раздела зависит от размера оперативной памяти: чем больше оперативки, тем меньше раздел swap. Если у тебя 1 Гб оперативки, можешь вообще отказаться от swap-раздела. При 512 Мб можно сделать swap-раздел в 128 Мб (вдруг не хватит). Если есть всего лишь 256 Мб, установи размер swap равным размеру ОЗУ, то есть тоже 256 Мб.

* Первичный/расширенный, Linux ext2/ext3 – для Linux. Linux все равно, с какого раздела загружаться – с первичного или расширенного. У меня это расширенный раздел. Ей так же все равно, будет ли этот раздел активным, в то время как для Windows 98 эти условия (первичный и активный) обязательны. Предлагаю сделать этот раздел размером 4 Гб. Для большинства дистрибутивов хватит, если, конечно, ты не будешь устанавливать все пакеты. Для твоих данных тоже хватит. Даже если у тебя винчестер на 120 Гб и 4 Гб кажутся каплей в море, не забудь, что Linux может использовать Windows-разделы, причем не только читать, но и записывать данные на эти разделы.

* Еще один раздел Linux (ext2/ext3), который используется в разных целях. Например, на него можно установить еще один дистрибутив, чтобы поэкспериментировать с ним. Можно хранить там пользовательские данные (/home), или, если 4 Гб не хватит, его можно смонтировать к /usr для хранения приложений. Размер этого размера установи по своему усмотрению. Минимум – 2 Гб (пригодится, если ты будешь устанавливать другой дистрибутив).

* Все остальное место – FAT32/NTFS-разделы. Хочешь – устанавливай сюда Windows XP или Windows 2000, хочешь – просто храни данные. Это будут диски D:, E: и т.д.

> При установке Linux обязательно создай загрузочный диск.

Теперь о том, как будем устанавливать. Предположим, ты хочешь установить двоих Windows и один Linux. Сначала нужно установить Windows 98 на первый (первичный, активный) раздел, то есть диск C:. Затем устанавливается Windows XP на один из FAT32/NTFS-разделов, например, диск D:. В последнюю очередь устанавливается Linux, которая установит загрузчик LILO/GRUB и обеспечит загрузку Windows и Linux. При установке Linux обязательно создай загрузочный диск – он поможет восстановить загрузчик Linux после очередной переустановки Windows. При установке Windows затирает MBR (Master Boot Record) и устанавливает в него свой загрузчик, поэтому двойная загрузка Windows и Linux становится невозможной. Если такое произошло, загружайся с загрузочного диска, созданного при установке Linux, регистрируйся как пользователь root и вводи команду Lilo. После этого перезагружай Linux (reboot) – и загрузчик восстанет из мертвых.

В случае если у тебя всего лишь две операционки, например, Windows XP и Linux, тогда все еще проще: сначала устанавливаешь XP (в любой раздел, ей все равно), а потом Linux, чтобы та установила свой загрузчик поверх загрузчика Windows. Linux сама определяет, что на компьютере установлена Windows, и настраивает соответствующим образом загрузчик – тебе остается только наслаждаться процессом. А вот более интересный случай. Устанавливаешь Windows XP, Linux Mandrake 10 и Linux Red Hat 7.3. Сначала, как обычно, устанавливается Windows XP. А потом более старый дистрибутив – Linux Red Hat 7.3, причем тут нужно отказаться от установки загрузчика. После этого устанавливаем Linux Mandrake 10 (дистрибутив поновее), и тут уже устанавливаем загрузчик в MBR. После этого редактируем файл /etc/lilo.conf и добавляем в него меню для загрузки Linux Red Hat:

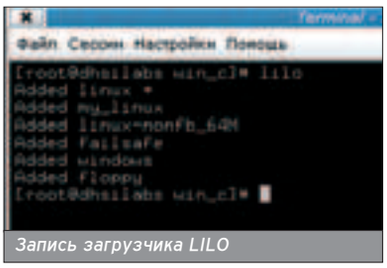
Фрагмент файла /etc/lilo.conf

```
image=/boot/vmlinuz
        label="my_linux"
        root=/dev/hda5
        initrd=/boot/initrd.img
        append="mem=256M "
        read-only
```

Предположим, Red hat 7.3 установлен на раздел /dev/hda5 и размер оперативной памяти равен 256 Мб. Перед записью загрузчика убедись, что ядро второго дистрибутива действительно называется так, как это указано в lilo.conf (/boot/vmlinuz). Если все правильно, нужно записать загрузчик:

```
lilo
```

Запись загрузчика LILO можно посмотреть на скрине.



Запись загрузчика LILO

ПРОСМАТРИВАЕМ WINDOWS-РАЗДЕЛЫ

В последнее время форматы файлов становятся более универсальными. Например, в Linux можно смотреть те же фильмы и слушать ту же музыку, как и в Windows. Благодаря пакету Open Office в Linux можно работать с документами MS Office. Не говоря уже о просмотре текстовых, .HTML- и .pdf-файлов. Особой нужды в изоляции двух этих систем нет. Наоборот, нужно настроить Linux, чтобы она смогла работать с файлами, расположенными на Windows-разделах. Для этого в файл /etc/fstab нужно добавить следующие строки (если за тебя это не сделала операционная система при установке):

Фрагмент файла /etc/fstab

```
/dev/hda1 /mnt/win_c vfat umask=0,iocharset=koi8-u,codepage=866 0 0
/dev/hda8 /mnt/win_d vfat umask=0,iocharset=koi8-u,codepage=866 0 0
/dev/hda9 /mnt/win_e vfat umask=0,iocharset=koi8-u,codepage=866 0 0
/dev/hda10 /mnt/win_f vfat umask=0,iocharset=koi8-u,codepage=866 0 0
```

Разберемся, что тут написано. Первый параметр – имя раздела, затем идет точка монтирования. Это означает, что Windows-раздел /dev/hda1 будет примонтирован к каталогу /mnt/win_c (по сути, это диск C; имя каталога можно указать по собственному усмотрению, например, /mnt/c), раздел /dev/hda8 будет примонтирован к /mnt/win_d и т.д. На то, что это Windows-разделы, указывает тип файловой системы, vfat (FAT32). Параметры кодировки iocharset=koi8-u,codepage=866 указываются отдельно для каждого раздела. Ты хочешь увидеть "Мои документы", а не "??? ?????????", правда? Последние два параметра (0 0) относятся к Linux-разделам, поэтому устанавливать их для Windows-разделов не нужно. После редактирования файла fstab выполнить команду mount -a или перезагрузить компьютер.

СОХРАНЯЕМ ФАЙЛЫ НА WINDOWS-РАЗДЕЛЕ

Разные дистрибутивы по-разному относятся к Windows-разделам. Одни вообще о них и знать не знают, поэтому добавлять информацию о них в файл /etc/fstab приходится вручную. Другие делают их доступными только для чтения - параметр ro, который нужно удалить, если хочешь записывать данные на раздел. Третьи напрочь забывают о кодировке, поэтому вместо родных букв выползают вопросительные знаки. А четвертые все делают правильно, но опять-таки забывают об одном параметре. О параметре umask, который указывает маску прав доступа при записи файла на раздел. Так как FAT32 не поддерживает прав доступа (а NTFS поддерживает права доступа, но не поддерживает umask), нужно указать umask=0, иначе каждый раз при записи на Windows-радел будешь получать сообщение о недопустимой операции – система будет пытаться установить права доступа к файлу, а эта операция недопустима для Windows-раздела.

Итак, чтобы записывать данные на FAT32-раздел нужно:

* удалить параметр ro в файле fstab, если такой есть;
* установить umask=0 для каждого Windows-раздела.

А вот для записи на NTFS-раздел понадобится еще и перекомпилировать ядро, потому что по умолчанию Linux только читает данные с NTFS-раздела. Стоит ли это делать? В ядро версии 2.6 включен так называемый безопасный драйвер записи на NTFS, который позволяет только перезаписывать файлы без изменения размера, но не позволяет создавать, модифицировать (чтобы изменялась длина), а также удалять файлы и каталоги. В ядре 2.5 был небезопасный драйвер, его использование часто приводило к потере данных на NTFS-разделе, поэтому он был заменен безопасной версией.

Можешь попробовать включить запись данных на NTFS. Могу предположить, что заголовочные файлы, необходимые библиотеки и компилятор gcc у тебя уже установлены. Перейди в каталог /usr/src/linux и выполни команду (все это от имени root):

```
# make xconfig
```

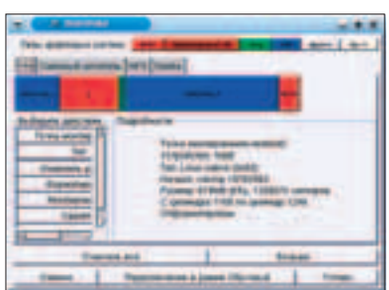
В разделе File Systems перейди в подраздел DOS/FAT/NT File systems и включи опцию NTFS write support. Заодно сможешь прочитать, что может, а что не может безопасный драйвер записи NTFS. После этого нужно перекомпилировать ядро. Введи команду

```
# make dep
```

После завершения ее работы ввести команду

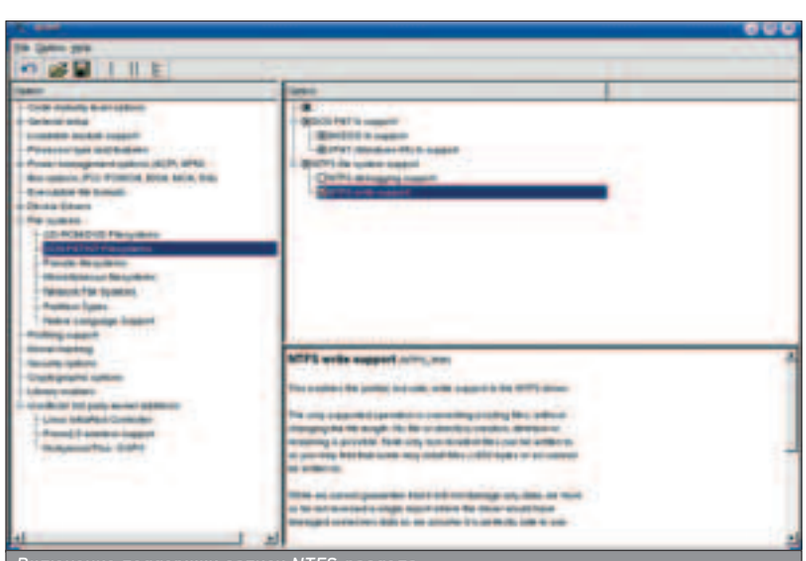
```
# make bzImage
```

Если исходники ядра и компилятор установлены корректно, то минут через 20 (это зависит от версии ядра и от быстродействия твоей системы) ты получишь откомпилированное ядро. Обычно оно помещается в каталог »



DiskDrake - аналог Partition Magic

Пользователь guest обязательно должен быть в системе.



Включение поддержки записи NTFS-раздела

/usr/src/linux/arch/i586/boot (или, например, i686 - это зависит от архитектуры твоего процессора). Теперь следует откомпилировать модули, которые будут использоваться ядром:

```
# make modules
```

И установить их:

```
# make modules_install
```

Перед установкой модулей сделай резервную копию модулей старого ядра (каталог /lib/modules). Теперь можно ввести команду

```
# make install
```

для установки только что созданного ядра, но я не рекомендую этого делать. Сначала нужно протестировать новое ядро: открой в любом редакторе файл /etc/lilo.conf :

```
# vi /etc/lilo.conf
```

Добавь в него следующие строки:

Фрагмент файла /etc/lilo.conf

```
image=/usr/src/linux/arch/i586/boot/bzImage
label=my_linux
# измени корневую ФС - у тебя она другая
root=/dev/hda5
append=" mem=128M"
read-only
```

Потом введи команду:

```
# lilo
```

Теперь перезагрузи систему:

```
# reboot
```

Попробуй загрузить ядро. В случае если всплывут ошибки, ты всегда сможешь загрузить старую версию.

СЕТЬ MICROSOFT

Попав в сеть Microsoft на своей Linux-машине, чувствуешь себя немного обделенным. Нет сетевого окружения, нельзя просмотреть, кто есть в сети (разве что пропинговать нужный компьютер, но для этого нужно помнить его IP), нельзя использовать общие диски и принтеры... В общем, неудобно как-то. И тут на помощь приходит программа LinNeighborhood и пакет Samba, который превращает твою Linux-систему в станцию сети Microsoft, причем так, что другие пользователи сети Microsoft не замечают этого. Что для этого нужно? Установить и настроить пакета Samba. В этой статье рассмотрю только базовую настройку пакета, а именно: настроим Linux-странцию, которая будет предоставлять в совместное использование один каталог. Если тебе интересна эта тема, более подробно можешь почитать о настройке Samba в моей книге "Linux-сервер своими руками" или просто в Сети - документов с описанием настройки Samba море. При установке пакетов обрати внимание на то, что пакет Samba-common должен быть установлен до установки пакетов Samba-client и Samba.

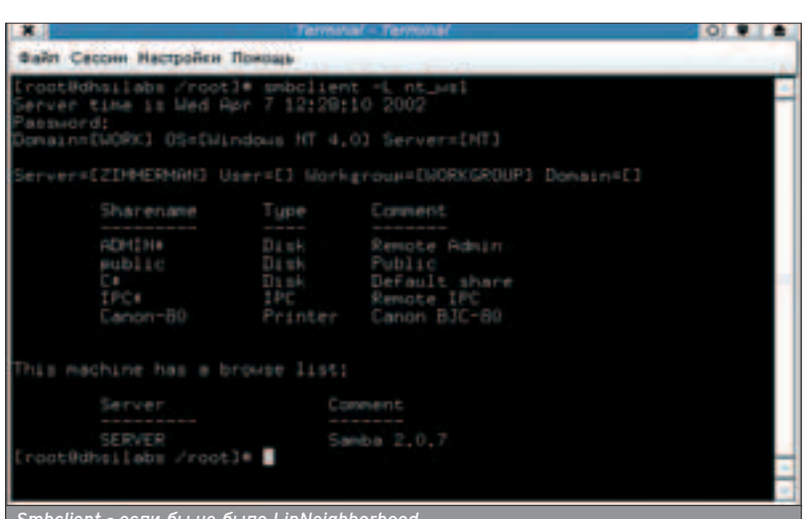
Первый пакет позволяет использовать общие ресурсы, а второй - предоставлять их в совместное использование. Обычно при установке Linux устанавливаются первые два пакета, а третий нужно ставить самостоятельно. Если ты хочешь использовать совместные ресурсы и предоставлять свои ресурсы в совместное использование, установи все пакеты. Отредактируй файл /etc/samba/smb.conf таким образом:

Файл /etc/samba/smb.conf

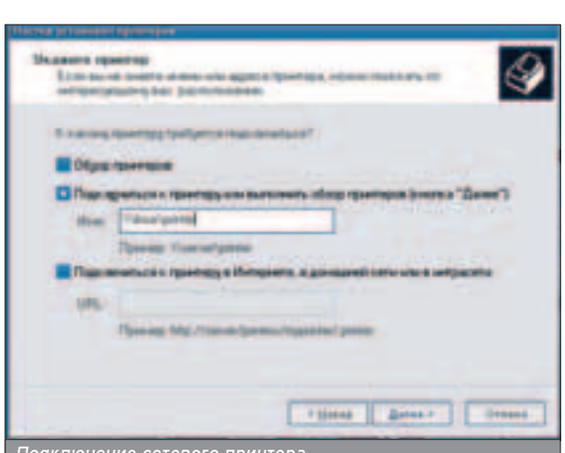
```
[globals]
workgroup = WORK
comment = Windows 98
guest account = guest
security = share
load printers = no
client code page = 866
character set = koi8-r
encrypt passwords = yes
socket options = TCP_NODELAY
dns proxy = no
wins support = yes
domain master = no

[homes]
comment = Home Resources
browseable = yes
writable = no

[public]
comment = Public Directory
path = /home/samba
read only = yes
```

Теперь разберемся, что было сотворено. В первых двух строках мы становимся членом группы WORK и маскируемся под Windows 98. Потом объявляем, что к нам можно подключаться, используя гостевую запись guest. Этот пользователь должен существовать в твоей системе! Добавь его: adduser guest. Это нужно для того, чтобы к компьютеру смогли обращаться другие пользователи. Помнишь ситуацию, когда после расшаривания ресурса на Windows XP к нему не мог подключиться ни один пользователь? А во всем виноват компьютер с Windows XP, в котором закрыт гостевой аккаунт. Уровень безопасности - share. В этом случае имя пользователя и пароль будут запрашиваться при каждом подключении к ресурсу. По умолчанию используется значение user, которое подразумевает, что будет использовано имя пользователя, под которым пользователь вошел в домен NT. Этот параметр (user) удобен для сервера, но не для рабочей станции.

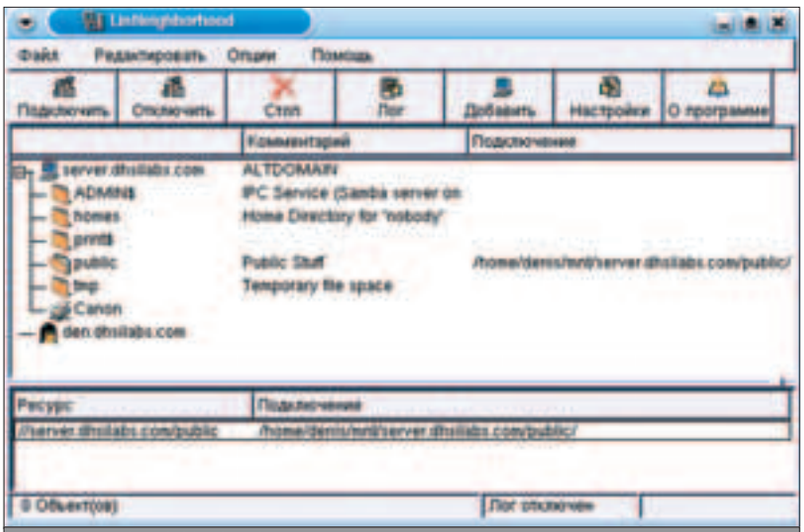
Следующая строка говорит о том, что мы не будем передавать свой принтер в общее пользование - пусть на своих печатают. Если хочешь все-таки сделать свой принтер достоянием общественности, нужно определить параметры printing, printcap name, а также секцию Printers, в которой ты определишь, какие принтеры можно предоставить в совместное использование, а также их параметры. Опция encrypt passwords = yes разрешает использование зашифрованных паролей, которые используют опера-



Smbclient - если бы не было LinNeighborhood



Подключение сетевого принтера



Программа LinNeighborhood

ционные системы Windows NT SP3, Windows 2000 и Windows 98. Если твоя версия Samba настолько старая, что не поддерживает эту опцию, установи более новую. Конечно, можно отключить шифрование паролей в реестре Windows, но это создаст больше проблем, чем установка новой версии Samba. Почему? Сколько Windows-машин в твоей сети? Одна, две, а может быть, двадцать - на каждой из них придется отключить шифрование паролей. Это тоже не отразится положительным образом на безопасности сети.

Опции client code page и character set обеспечивают корректную работу с русскими именами файлов. Назначение других параметров ты узнаешь в справочной системе. Пока ты ее не прочитал и не знаешь, зачем нужна опция domain master, не включай ее, а то станешь контроллером домена. Правда, чтобы контролировать домен, включения этой опции недостаточно - нужно настроить еще кое-что. Секция homes определяет параметры совместно используемых ресурсов, разрешает просматривать их через "Сетевое окружение" (browseable = yes) и запрещает что-либо писать в них (writable = no). Секция public определяет один общий каталог - /home/samba. К нему могут подключаться все кому не лень, но все они могут только читать его. После этих настроек перезапусти сервис smb:

```
/etc/init.d/smb restart
```

Доступ к совместно используемым ресурсам осуществляется с помощью программ smbclient и smbmount. Как их использовать, ты можешь прочитать в справочной системе. Скажу сразу: их использовать не очень удобно. Конечно, если ты работаешь в текстовом режиме и система X Window не установлена - другого выхода нет. Установи программу LinNeighborhood (пакет и имя команды для запуска называются так же). Запусти ее и наслаждайся!

В верхней части окна отображаются все узлы в сети, а в нижней - подключенные в данный момент общие ресурсы. В сети находятся две машины. Первая - это контроллер домена server.dhsilabs.com, а вторая - это моя машина den.dhsilabs.com.

Сервер расшарил такие ресурсы:

1. ADMIN$;
2. Homes;
3. Print$;
4. public;
5. tmp;
6. принтер Canon.

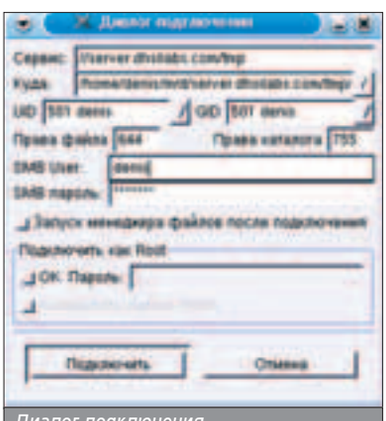
Ресурс public в данный момент подключен к каталогу /home/denis/mnt/server.dhsilabs.com/public, с которым можно работать как с обыкновенным каталогом файловой системы. Подключение обычно производиться к каталогу: /home/<имя_пользователя>/mnt/<имя_сервера>/<имя_ресурса>. Чтобы подключить ресурс, выдели его и нажми кнопку "Подключить" или дважды щелкни на нужном ресурсе. Появится окно "Диалог подключения", в котором нужно указать определенные параметры:

Обычно бывает достаточно указать имя пользователя и пароль, если они вообще нужны - можно подключаться с помощью гостевой записи к общему каталогу. Кнопка "Добавить" присоединяет к коллективу любимую машину, то есть машину, которую ты часто используешь. Совсем не обязательно, чтобы машина находилась в одной рабочей группе с тобой.

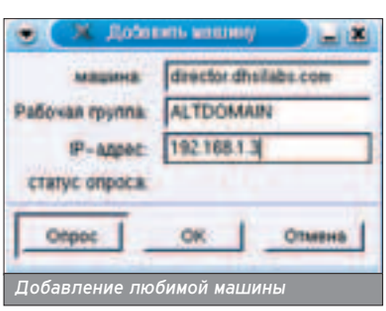
Нажав на кнопку "Настройка", можно определить параметры программы, но предлагаемые параметры вполне приемлемы для большинства пользователей. Единственное, что нужно указать, так это имя рабочей группы.

Для сканирования всей сети можно использовать команду меню "Опция"-> "Просмотреть всю сеть".

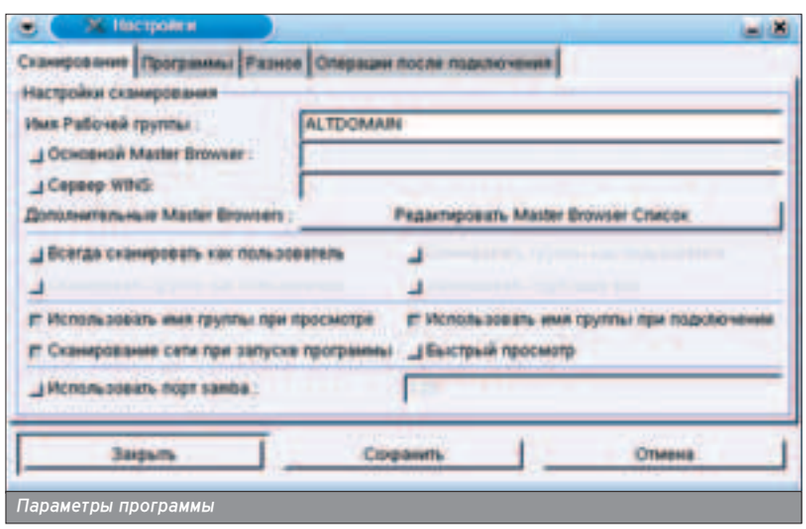
Надеюсь, эта статья по-настоящему укрепит дружбу Windows и Linux на твоей машине. Да здравствует дружба!



Диалог подключения



Добавление любимой машины



Параметры программы

Крис Касперски aka мыщъх

ПРОФЕССИИ ЭМУЛЯТОРОВ

ВИРТУАЛЬНЫЕ МАШИНЫ ПОД *NIX И НЕ ТОЛЬКО

Виртуальная машина – великая вещь! Кем бы ты ни был: продвинутым пользователем, администратором, программистом или даже воинственным хакером, – эмулятор тебя выручит и всегда поможет. Весь вопрос в том, когда и как. Вот об этом мы и собираемся рассказать!

Эмуляторы прочно вошли в нашу жизнь и не собираются из нее никуда уходить. Наоборот, их поголовье увеличивается с каждым днем. Мы не будем рекламировать каких-то конкретных представителей этого вида – эмулятор своей мечты каждый может найти и самостоятельно (заходим в Google, говорим ему "обзор эмуляторов" или что-то в этом роде, щелкаем "мне повезет"). Лучше мы расскажем, что с этим самым эмулятором можно сделать, то есть как правильно его употребить.

ПОЛЬЗОВАТЕЛИ

Вообрази себе картину: ты прочитал в компьютерном журнале о замечательной игрушке, полюбил ее всеми фибрами своей души и вдруг обнаружил, что на твоей оси она не идет. Прямо как обухом по голове и ножом в спину! Хуже всего приходится пользователям FreeBSD – игр под нее днем с огнем не найдешь. Для Windows места не жалко, но перезагружаться каждый раз, чтобы запустить игру, – нет уж, увольте! А если это игра под Mac или Sony Playstation? Современные аппаратные мощности заставляют забыть о родном "железе" и эмулировать весь компьютер целиком, открывая безграничный мир программного обеспечения. Теперь ты уже не привязан к какой-то конкретной платформе и можешь запускать любые программы независимо от того, для какого компьютера они были написаны – ZX SPECTRUM или X-Box. Главное – найти хороший эмулятор!

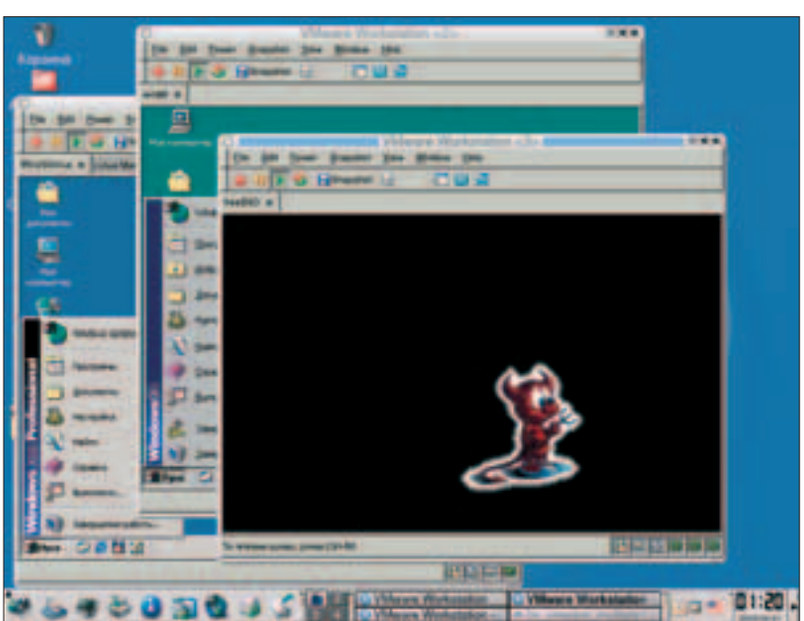
Основная операционная система становится как бы фундаментом для целого зоопарка гостей. Одну из клеток этого зверинца можно (и нужно!) выделить под своеобразный карантин-отстойник. Известно, что при установке новой программы ты всегда рискуешь уронить операционную систему, – кривой инсталлятор, конфликт библиотек, Add-Ware или просто карма у нее такая. Программы, полученные из ненадежных источников, лучше держать подальше ото всех остальных. Просто выдели им отдельную виртуальную машину в эмуляторе, и они оттуда не вырвутся!

АДМИНИСТРАТОРУ

Для администраторов эмулятор – это в первую очередь полигон для всевозможных экспериментов. Поставь себе десяток различных *nix-систем и издевайся над ними по полной программе. Устанавливай систему, сноси ее и снова устанавливай, слегка подкрутив конфигурацию. На работу ведь принимают не по диплому, а по специальности, а специальность приобретается только в боях. То же самое относится и к восстановлению данных. Без специальной подготовки Disk Editor на своей рабочей машине лучше всего не запускать, а Disk Doctor – тем более. Нет никакой гарантии, что он действительно вылечит диск, а не превратит его в винегрет. Короче говоря, эмулятор – это великолепный тестовый стенд, о котором раньше нельзя было даже мечтать.

В больших организациях администратор всегда держит на резервном компьютере точную копию сервера и все заплатки сначала обкатывает на нем. В конторах поменьше отдельной машины под это дело никто не даст и приходится прибегать к эмулятору. На нем же тестируются различные эксплойты, и если факт уязвимости подтверждается, принимаются оперативные меры по ее устранению.

Общение виртуальной машины с основной операционной системой и другими виртуальными машинами обычно осуществляется через локальную сеть. Виртуальную, разумеется. При наличии 512-1024 Мб памяти можно создать настоящий корпоративный интранет – с SQL и web-серверами, DMZ-зоной, брандмауэром и несколькими рабочими станциями, свободно умещающимися внутри домашнего компьютера. Лучшего полигона для обучения сетевым премудростям и не придумаешь. Хочешь – атакуй, хочешь – администрируй.



Неважно, какая у тебя ось! Важно – какой у тебя эмулятор!

РАЗРАБОТЧИКАМ

Больше всего эмуляторы любят разработчики драйверов. Ядро не прощает ошибок и жестоко разрушает жесткий диск, уничтожая все данные, накопленные за многие годы. Перезагрузки и зависания – вообще обычное дело, к которому привыкаешь, как к стуку колес или шороху шин. К тому же большинство отладчиков ядерного уровня требует наличия двух компьютеров, соединенных COM-шнурком или локальной сетью. Для профессионального разработчика это не роскошь, но... куда их ставить? Окружишь себя мониторами, а потом как дурак крутишь во все стороны головой – отвались моя шея!

С эмулятором все намного проще. Ни тебе потери данных, ни перезагрузок, а всю работу по отладке можно выполнять на одном компьютере. Естественно, совсем уж без перезагрузок дело не обходится, но пока перезагружается виртуальная машина, можно делать что-то полезное на основной (например, править исходный код драйвера). К тому же можно заставить эмулятор писать команды в лог и потом подсмотреть, что довело драйвер до смерти (правда, не все эмуляторы это умеют).

В GENETIC-ядре FreeBSD отладчика нет, а отладочное ядро вносит в систему побочные эффекты. Windows-отладчики ведут себя похожим образом и окончательное тестирование драйвера должно проходить в "безлошадной" конфигурации, начисто лишая разработчика всех средств отладки и мониторинга.

А что прикладные программисты? Эмуляторы позволяют держать им под рукой всю линейку операционных систем, подстраивая свои программы под особенности поведения каждой из них. У Windows всего две системы – NT плюс 9x, да и то у них голова кругом идет, а *nix-системы намного более разнообразны!

Все коварство багов в том, что они имеют склонность появляться только в строго определенных конфигурациях. Установка дополнительного программного обеспечения, а уж тем более перекомпиляция ядра может их спугнуть, и тогда ищи-свищи. А это значит, что до тех пор пока баг не будет найден, ничего менять в системе нельзя. На основной машине выполнить это требование затруднительно, зато легко на эмуляторе! Виртуальная машина, отключенная от сети (в том числе и виртуальной), в заплатах не нуждается. Но как же тогда обмениваться данными? К твоим услугам – дискета и CD-R.

Самое главное – эмуляторы позволяют создавать "слепки" состояния системы и возвращаться к ним в любое время неограниченное количество раз. Это значительно упрощает задачу воспроизведения сбоя (то есть определения обстоятельств его возникновения). Чем такой слепок отличается от дампа памяти, сбрасываемого системой при сбое? Как и следует из его названия, дамп включает в себя только память, а "слепок" – все компоненты системы целиком (диск, память, регистры контроллеров и т.д.).

Разработчики сетевых приложений от эмуляторов вообще в полном восторге. Раньше ведь как: ставишь второй компьютер, сажаешь за него жену и долго и нудно объясняешь, какие клавиши ей нажимать. Теперь же отладка сетевых приложений упростилась до предела.

ХАКЕРАМ

Эмулирующие отладчики появились еще во времена MS-DOS и сразу же завоевали бешеную популярность. Неудивительно! Рядовые защитные механизмы применяют две основные методики для борьбы с отладчиками – пассивное обнаружение отладчика и активный захват отладочных ресурсов, делающий отладку невозможной. На эмулирующий отладчик эти действия никак не распространяются: он находится ниже виртуального процессора (поэтому для отлаживаемого приложения совершенно невидим) и не использует никаких ресурсов эмулируемого процессора.

Слепки системы очень помогают при взломе программ с ограниченным сроком использования. Ставим программу, делаем слепок, переводим дату, делаем еще один слепок. Смотрим, что изменилось. Делаем выводы и отламываем от программы лишние запчасти. В урезанной редакции эта методика выглядит так: устанавливаем защищенную программу на отдельную виртуальную машину. Делаем "слепок". Все! Защите хана! Сколько бы ни запускали "слепок", она будет наивно полагать, что запускается в первый раз. Не сможет она привязываться и к оборудованию – оборудование эмулятора не зависит от аппаратного окружения, предоставляя нам неограниченную свободу выбора последнего.

Попутно эмулятор освобождает от необходимости ставить ломаемую программу на свою основную машину. Во-первых, некоторые программы, обнаружив, что их ломают, пытаются как-то напакостить на винчестере, а если даже и не напакостят, то как пить дать сглючат. Так пусть лучше глючит на эмуляторе – это во-вторых.

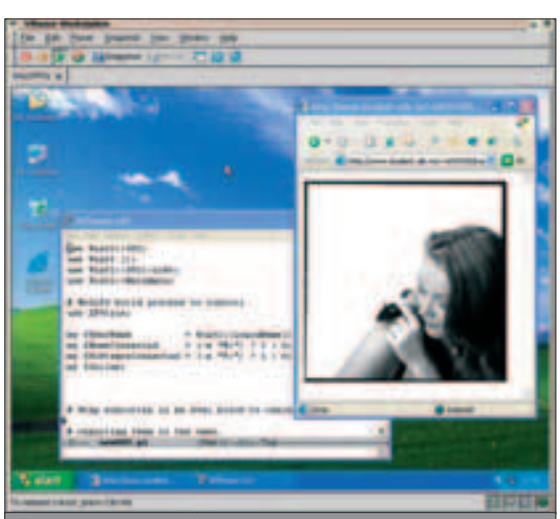
ЗАКЛЮЧЕНИЕ

Эмуляторы преображают компьютерный мир, с каждым днем наращивая потенциал своих возможностей. На мощнейших Pentium'ах даже DOOM III эмулируется с приемлемой скоростью, а это значит, что время поголовной установки эмуляторов уже пришло. Стоит только попробовать, и ты уже ни за что не откажешься от десятка своих любимых виртуальный машин, позабыв про основную платформу.

КАК НАСТРОИТЬ SOFTICE

При попытке использования SoftIce под Windows 2000, запущенной из-под VMWare начинаются сплошные лапти: SoftIce работает только из text-mode режима, развернутого на весь экран (заходим в FAR, жмем <ALT>+<ENTER>, затем <CTRL>+<D>), а во всех остальных режимах наглухо завешивает систему. Кстати, под Windows 98 он чувствует себя вполне нормально, но переход на Windows 98 – не вариант.

Это известный глюк Ice'a, признанный NuMega и устраненный лишь в Driver Studio версии 3.1 (в официальной формулировке это именуется "поддержкой VMWARE"). Подробности можно найти в документации (см. \Compuware\DriverStudio\Books\Using SoftICE.pdf, приложение E – SoftIce and VMware). При этом в конфигурационный файл виртуальной машины (имя_виртуальной_машины.vmx) необходимо добавить строку svga.maxFullscreenRefreshTick = "2" и vmmouse.present = "FALSE". Мышь работать не будет, да она в SoftIce не сильно кому и нужна.



Отладка прикладной программы под эмулятором

Ляхов Николай aka r.o.o.t (r.o.o.t@bk.ru)

X-ОКОШКИ

ГРАФИЧЕСКАЯ СИСТЕМА LINUX ПОД ПРИЦЕЛОМ

Если ты терпеть не можешь командную строку, то для тебя на первый план выходит графический интерфейс твоей системы. Давай поговорим о различных реализациях графического интерфейса Linux и об оконных менеджерах.

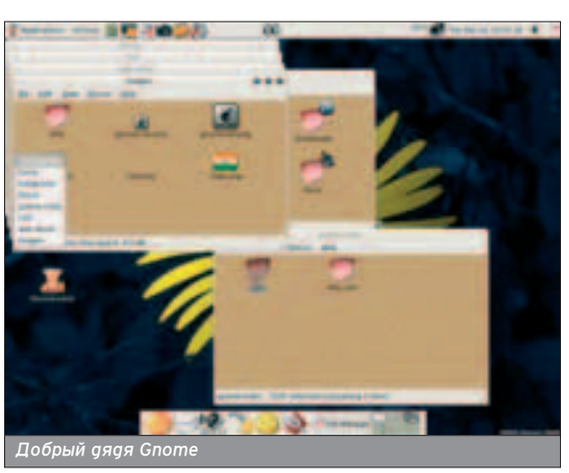
X WINDOW SYSTEM

Почти у всех моих друзей, сидящих под Linux, первой командой было... Startx. Вот про X Window System мы сегодня и поговорим.

Сейчас через X Window System можно отконфигурировать все, что может понадобиться для нормальной работы в Linux. Я даже видел в каком-то дистрибутиве программку под X Window System, позволяющую поставить WinModem одним кликом мыши! Красота! X Window System - это графический пользовательский интерфейс (Graphical User Interface) для Linux. Расскажу о наиболее распространенных оконных менеджерах для Linux - Gnome и KDE, а потом и об альтернативе им.

GNOME (GNU NETWORK OBJECT MODEL ENVIRONMENT)

Сетевая Объектная Среда (GNU). Классика для Linux. Сейчас используется третья версия программы. Gnome то называли лучшей, то опускали ниже системника. Еще недавно KDE сильно обгонял этот оконный менеджер, но энтузиасты взялись за ум и стали улучшать свое детище. Кроме того, его поддерживают многие дистрибутивы Linux (часто именно этот менеджер установлен по умолчанию). Gnome имеет относительно неболь-



Добрый дядя Gnome

шие размеры и работает довольно шустро. Ему по традиции приписывают сложность настроек, особенно для новичков. Но это не такая уж проблема. А проблема, к сожалению, есть! Плоховато у него с языком, на котором "разговаривал Ленин"... Английский - это, конечно, хорошо, но меня, например, учить какие-то термины, кроме компьютерных, напрягает. Чуть не забыл: в рунете сложно найти стоящую документацию по этой системе! Не очень впечатляет и набор поставляемых с Gnome программ (правда, их легко скачать в интернете).

KDE (K DESKTOP ENVIRONMENT)

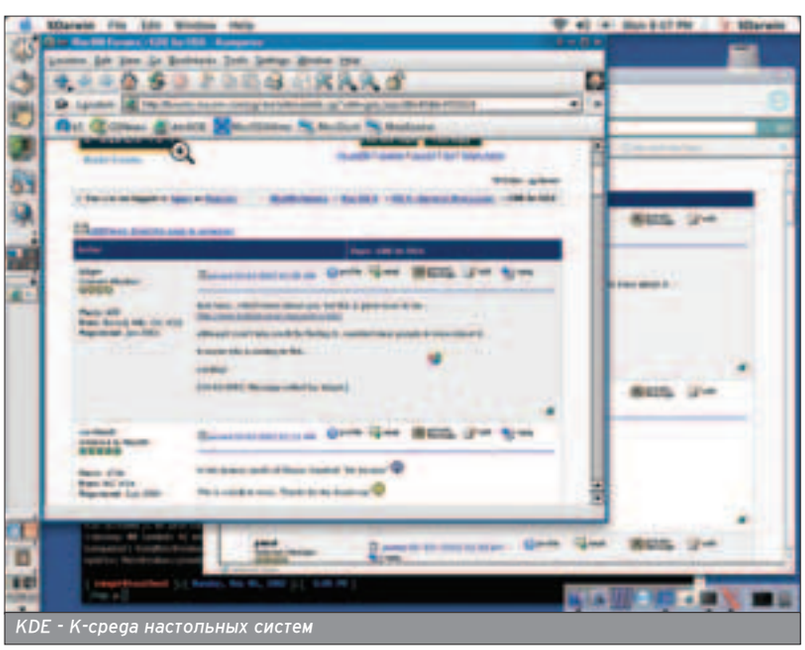
K - среда настольных систем. Не мила ярым линуксоидам за то, что подвержена разным влияниям. Но надо признать, что эти влияния (Windows и MAC OS) сделали свое доброе дело. Получившийся менеджер довольно успешно щемит конкурентов и даже порой обгоняет Gnome по количеству пользователей. На сегодня актуальна KDE3. Говорят, что KDE больше похожа на Windows, нежели GNOME, и из-за этого его рекомендуют новичкам, но, по-моему, можно найти соответствующую тему и под Gnome, так что это не так важно. Вся изюминка KDE именно в наборе специфических программ из его состава, хотя никто не мешает запускать и "неродные" программы. Даже "Офис" здесь свой - называется K-office. У него много фанатов, и он сильно отличается от похожих друг на друга внешне Open Office и MS Office. Настраивать KDE так же просто, как Windows 3.11!

Все, о них больше не буду, потому что в этом номере есть материал с сравнением этих гигантов... :)

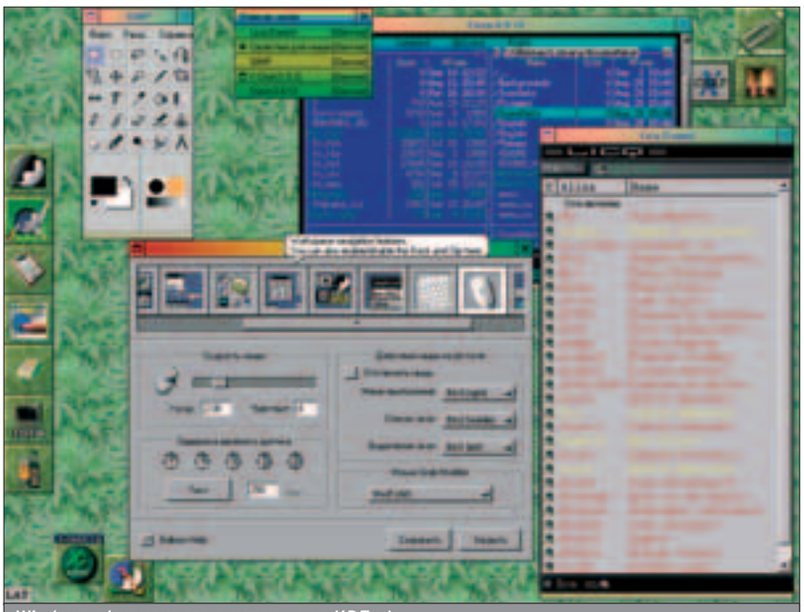
Давай рассмотрим альтернативные менеджеры. О трех из них подробнее всего Windowmaker, XFCE, Fluxbox.

WINDOWMAKER

Взять последнюю версию можно с ftp://windowmaker.org/pub/source/release/WindowMaker-0.91.0.tar.gz. Возможно, ты еще помнишь NextStep - знаменитую систему начала 90-х годов, которая была призвана явить миру графический интерфейс будущего. Попытка эта в силу ряда



KDE - K-среда настольных систем

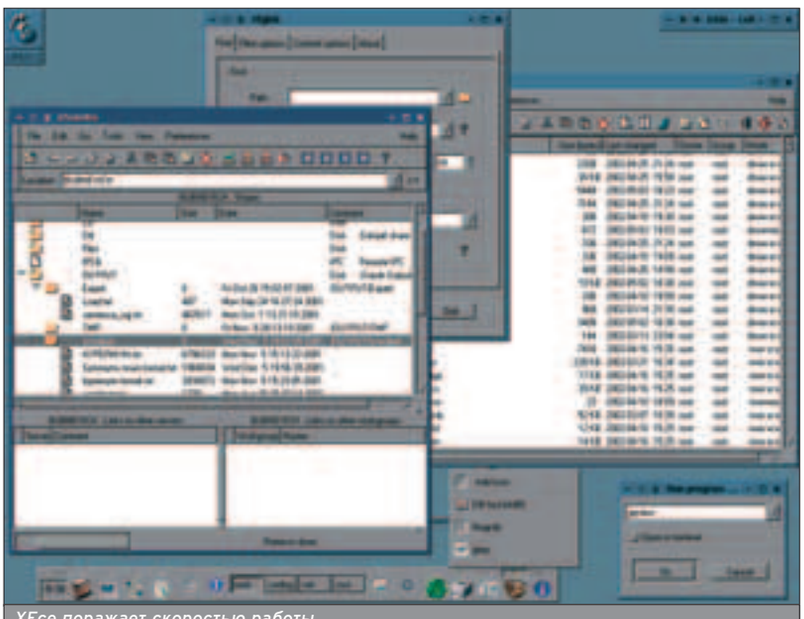


Windowmaker намного симпатичнее KDE ;-)

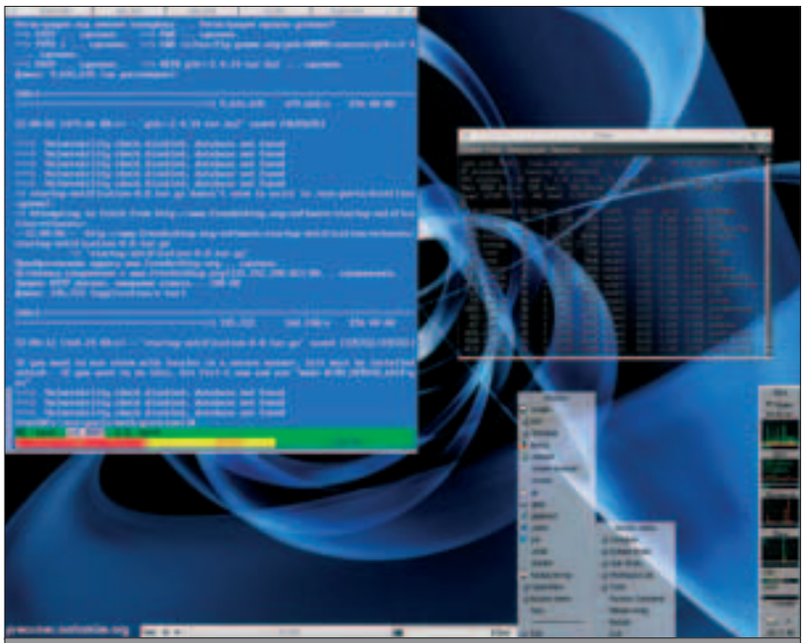
причин не удалась (хотя в некоторых академических учреждениях России эта система до недавнего времени с успехом использовалась на собственной аппаратной платформе ;)), но дело ее не пропало: она оказала влияние на многие графические среды Unix-систем, как коммерческие, так и открытые. А в мире XFree86 она получила дальнейшее развитие в виде оконных менеджеров AfterStep и Windowmaker. Имеет принципиально отличный от Windows внешний вид (в связи с этим - несколько непривычен), предоставляет неплохие возможности, легко и гибко конфигурируется. Пользуется (заслуженно) довольно большой популярностью. Возможных настроек достаточно для удовлетворения личных эстетических идеалов и для внесения разнообразия в производственный процесс. Устойчивость - на уровне (я с зависаниями практически не сталкивался). Основные манипуляции с приложениями выполняются (при наличии минимальной привычки и после несложных настроек) легко и быстро. Тормозом Windowmaker не назовешь! По крайней мере, при использовании на машине с процессором класса P-III и 128 Мб памяти. Windowmaker позволяет запускать все рассчитанные на KDE приложения, обладает при этом более привлекательной внешностью, поэтому эти два продукта можно считать полноценными соперниками. На мой взгляд, Windowmaker - лучшее соотношение дизайн/качество :).

XFCE

Скачать программу можно тут: www.xfce.org/index.php?page=download&lang=en. XFce The Cholesterol Free Desktop Environment - также интегрированная (то есть содержащая специфичные для нее приложения и утилиты), имеющая свой собственный оконный менеджер графическая среда, основанная на библиотеке Gtk. Впрочем, она стандартно



XFce поражает скоростью работы



После нескольких дней работы ты влюбишься во FluxBox

входит во многие дистрибутивы Linux. "Xfce is a lightweight desktop environment for various *NIX systems. Designed for productivity, it loads and executes applications fast, while conserving system resources. XFce - это компактная рабочая среда для систем *NIX. Она была разработана для обеспечения максимальной продуктивности, и поэтому быстрее загружает и исполняет приложения, не занимая при этом много системных ресурсов.)," - сказал однажды Оливер Фордан (Olivier Fourdan), создатель XFce. Первый же запуск XFce показывает оправданность ее названия: все работает весьма быстро и при этом поражает своей импульсивностью. Поведение системы определяется ее собственным оконным менеджером, который включает следующие интерфейсные элементы: переключатель (Pager, в терминологии авторов) виртуальных экранов в верхнем правом углу и главную панель (XFce Main Panel) - внизу по центру. Вот вроде бы и все его основные отличия. Однозначно, самый быстрый менеджер.

FLUXBOX

FluxBox является одним из потомков семейства BlackBox. BlackBox был простым оконным менеджером без всяких особенностей. Причиной создания FluxBox стал дизайн BlackBox, который многим пришелся не по душе. Людям хотелось иметь более симпатичную графическую оболочку с приличным набором функций. Так появились два проекта: вышеупомянутый FluxBox и OpenBox, в основу которых был положен исходный код BlackBox. Наибольшей популярностью из них пользовался FluxBox, последнюю версию которого можно скачать на http://prdownloads.sourceforge.net/fluxbox/fluxbox-0.9.9.tar.gz.

Внешний вид FluxBox унаследовал от BlackBox небольшой набор графических тем. Хотя их штатного количества хватает примерно на два месяца: потом уже мозолит глаза. Но эту проблему можно решить или нарисовав новую тему самостоятельно, или стянув у кого-нибудь готовую. Если хочешь найти побольше тем, ищи здесь: http://themes.freshmeat.net. Несмотря на то, что темы разбиты на отдельные группы для FluxBox и BlackBox, они подходят как для одного менеджера, так и для другого.

О новшествах FluxBox. Реализованы закладки в меню рабочего стола. Для сравнения: закладки в браузере позволяют одновременно открыть несколько страниц в одном окне, а закладки Fluxbox дают возможность удобно(!) сгруппировать несколько окон на рабочем столе. Все окна в группе имеют одинаковые размеры и расположены строго одно под другим. Пробивает на ностальгию по Win 3.11... Для переключения на какое-либо из них достаточно навести курсор мыши или щелкнуть (в зависимости от настроек) по соответствующей закладке. Например, если тебе приходится работать сразу с несколькими почтовыми клиентами, то удобнее будет совместить их в одну группу и переключаться между ними при необходимости. И ты всегда будешь знать, где расположено каждое окно. В общем, все это словами не опишешь... После нескольких дней работы с FluxBox ты в него влюбишься и уже не будешь представлять своей жизни без него. Помимо закладок, во FluxBox реализованы функции, не входящие в стандартную поставку BlackBox, перечислять которые не имеет смысла, так как ты обнаружишь их почти сразу в процессе работы с FluxBox. Очень удачная программа.

ЕСЛИ ТЫ УСТАЛ ОТ X-МОНСТРОВ

На сайте fp.nsk.fio.ru я нашел интересную табличку сравнения этих оболочек: измерения производились в следующей конфигурации: P-III/533 (не Coppermine, 133Mhz шина, 512 Кб кеша), системная плата MSI-6163 (Intel BX), 128 Мб памяти (один модуль PC-133, неизвестного происхождения), Matrox G-400 с 16 Мб памяти, диск Quantum Fireball 8,4 Гб (ATA-66, 5400 об/с); прочие компоненты я считаю несущественными. Видеорежим - 1024*768 при 16-битном цвете. Сначала измерялось время запуска оконной среды из программы WMSelect по щелчку на соответствующей пиктограмме; фон - по умолчанию в каждой среде, за исключением Gnome, без использования фоновых рисунков. Для Gnome оказалось, что время его запуска сильно зависит от настроек Enlightenment, используемого в моей версии в качестве оконного менеджера; поэтому для последнего была подобрана достаточно типичная тема без архитектурных излишеств - в противном случае результат был бы на много хуже. Затем в каждой из сред измерялось время запуска прикладных программ, основанных на Qt (Webmaker и Klyx) и на Gtk (Bluefish и GIMP). Результаты измерений приведены в таблице (время в секундах, среднее из пяти измерений).

Ну вот, собственно, и все. А вообще - используй командную строку! Немного опыта - и ты от нее никогда не откажешься!

Время запуска Gnome очень сильно зависит от его настроек.

Среда	KDE	GNOME	WMaker	XFce	IceWM	FLWM
Запуск	15.60	4.00	1.09	1.78	2.09	1.00
WebMaker 2.37	1.63	1.25	1.28	1.03	1.03	
Klyx	2.03	2.03	1.50	1.10	0.82	0.97
Bluefish 2.56	2.31	2.10	1.59	1.50	1.66	
GIMP	5.69	3.30	2.31	2.20	2.07	1.90
Среднее 5.65	2.65	1.65	1.59	1.50	1.31	
Ср. для Qt 2.20	1.83	1.38	1.19	0.93	1.00	
Ср. для Gtk	4.13	2.81	2.21	1.90	1.79	1.78

Сравнение быстродействия графических сред

ТОВАРЫ В СТИЛЕ

ПРИСОЕДИНЯЙСЯ!

ЭКСКЛЮЗИВНАЯ КОЛЛЕКЦИЯ ОДЕЖДЫ И АКСЕССУАРОВ ОТ ЖУРНАЛОВ ХАКЕР И ХУЛИГАН

* Футболки, толстовки, куртки, бейсболки,
* Кружки, зажигалки, брелки,
* Часы и многое другое

Тел.: (095) 928-0360
(095) 928-6089
(095) 928-3574
www.gamepost.ru

GAMEPOST

Денис Колисниченко, dhsilabs@mail.ru

COUNTER STRIKE ПОД LINUX

ПОДНЯТИЕ ИГРОВОГО СЕРВЕРА

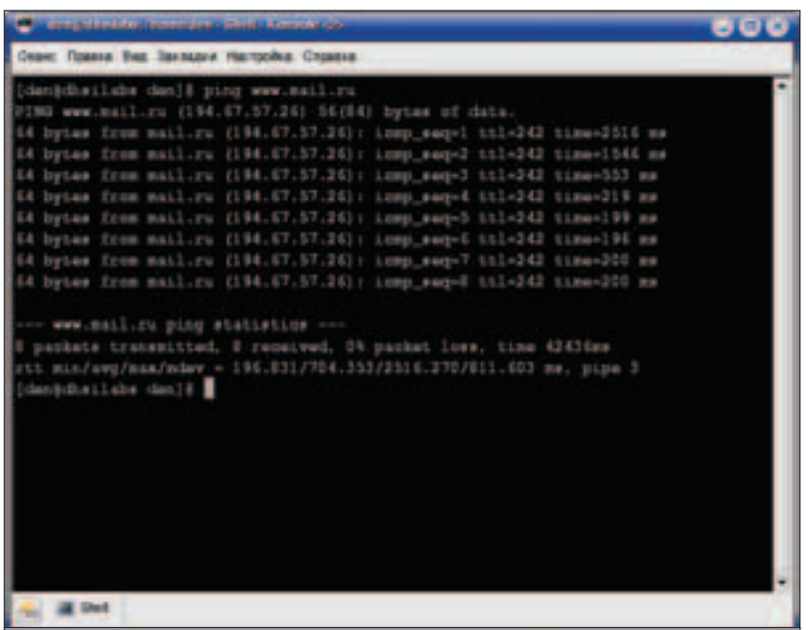
Ты можешь использовать свой домашний компьютер под управлением операционной системы Linux как угодно. Это может быть просто десктоп или, например, твое персональное хранилище файлов, то есть FTP-сервер. Но есть кое-что намного веселее! Поговорим о том, как можно превратить твою домашнюю машину в игровой сервер для какой-либо сетевой игры на примере известного хита - Counter Strike.

Итак, поднятие игрового сервера под Linux на примере игры Counter Strike. Что это даст именно тебе? Во-первых, твоя машина теперь уже не будет периодически "подтормаживать", когда ты играешь со своими соседями в эту игру. Ведь подумай: твоей бедной машинке приходится не только выполнять Counter Strike, но еще и обслуживать всех твоих соперников. А так у тебя будет выделенный сервер для игры, а тебе еще и спасибо скажут. Конечно, все это верно лишь при условии, что играть ты будешь уже на другом компьютере, на котором будет установлена Windows :-). Самое замечательное то, что с помощью Linux и игрового сервера Counter Strike ты можешь дать вторую жизнь своему старенькому компьютеру, а на новом запускать саму игру. Все дело в том, что игровой сервер не требует ничего сверхъестественного от твоего железа.

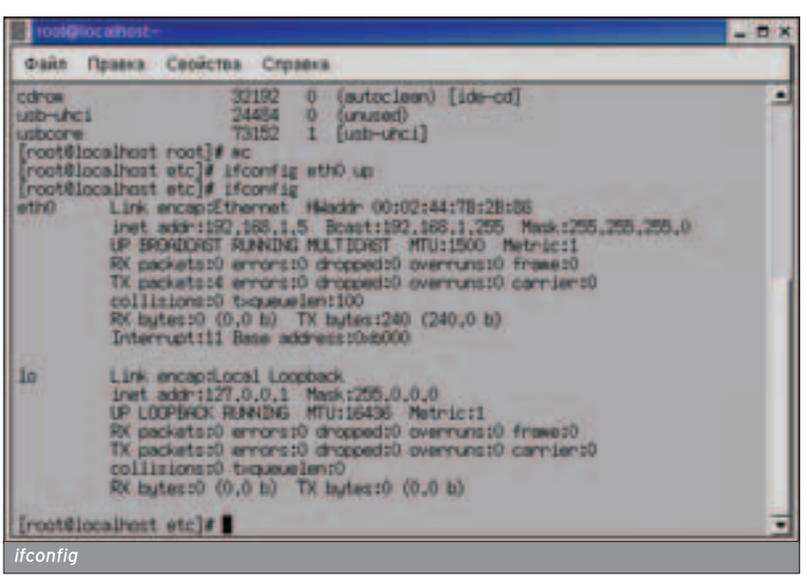
УСТАНОВКА ИГРОВОГО СЕРВЕРА

Установку Linux и настройку сети я описывать не буду - будем считать, что у тебя до этого все настроено и все работает. Чтобы убедиться в этом, запусти программу ping и пропингуй какой-нибудь компьютер. Если ping прошел, значит, сеть у тебя работает. А если нет? Не паникуй: может, тот компьютер, который ты пингуешь, просто выключен. Проверяется очень просто - пропингуй другой компьютер :-). Если эти компьютеры включены, а ping не проходит, значит, нужно проверять настройки сети. Если ты точно помнишь, что сеть настраивал, а она не работает (случай, что ты неправильно ее настроил, не рассматривается :)), запусти программу ifconfig - она тебе все расскажет и покажет. На современных дистрибутивах (если не отключена одна замечательная опция) сетевой интерфейс не работает, если повреждена физическая среда передачи данных, например, сетевой кабель или если просто этот кабель не подключен к компьютеру. Если у тебя модемное соединение, проверь сначала сигнал в линии, а потом уже настройки модема. Если же витая пара, проверь, не поврежден ли кабель. Нужно проверить и его обжимку: возможно, ты просто неправильно его обжал. Думаю, теперь твоя сеть заработала и можно двигаться дальше.

Для Counter Strike версии 1.3 тебе понадобятся следующие файлы:



Ping



ifconfig

```
hlds_l_3108_full.tar.gz
hlds_l_3108c_perf.tar.gz
cs_13_full.tar.gz
```

Первый файл – это выделенный сервер Half Life версии 3.1.0.8. Второй – патч-оптимизатор для этого сервера. Третий – выделенный сервер Counter Strike. Надеюсь, ты понимаешь, что устанавливать придется все эти три файла. Скачать файлы можно по адресу server.counter-strike.net. Я рекомендую скачивать именно оттуда, потому что на этом сервере постоянно выкладывают самые новые версии. Заходим под пользователем root, создаем в корне каталог /halflife, помещаем туда наши файлы:

```
su root
cd /
mkdir halflife
cd halflife
```

После того как ты скопируешь в этот каталог все эти файлы, их нужно будет распаковать:

```
tar xvzpf hlds_l_3108_full.tar.gz
```

По окончании работы архиватора выполни следующие команды:

```
move cs_13_full.tar.gz hlds_l/
cd /halflife/hlds_l
tar xvzpf cs_13_full.tar.gz
```

Первая перемещает сервер Counter Strike в каталог hlds_l, вторая переходит в этот каталог, а третья распаковывает файл cs_13_full.tar.gz в каталог hlds_l. Архив будет распакован в каталог cstrike. Теперь сервер необходимо настроить. Приступим. Первым делом открываем файл hlds_l/cstrike/motd.txt и пишем в нем сообщение для соседей, в котором напоминается, что за настройку сервера тебе полагается много пива. Потом можно подправить файл настроек сервера hlds_l/cstrike/server.cfg. Но пока я бы не стал этого делать. Просто открой его и посмотри, какие служебные переменные используются (обращай внимание также на их значения). В файле hlds_l/cstrike/mapcycle.txt прописываются карты, которые будут включены в mapcycle. В общем, на этом настройка и заканчивается. Впрочем, ты еще не раз вернешься к этим файлам.

ЗАПУСК СЕРВЕРА

И самое интересное - запуск нашего сервера. Перейди в каталог /halflife/hlds_l/. Потом создай файл cstrike_server_start в любом текстовом редакторе. Содержание этого файла будет таким:

```
#/bin/bash/
export
LD_LIBRARY_PATH=/halflife/hlds_l:$LD_LIBRARY_PATH
./hlds_run -game cstrike +ip your.ip.here +maxplayers
12 +map cs_assault
```

Вместо your.ip.here нужно вставить твой IP-адрес. Но это только в том случае, если ты хочешь, чтобы твой сервер был доступен через интернет. Для локальной сети опция +ip не используется. Теперь разреши этому файлу запускаться:

```
chmod +x cstrike_server_start
```

Вот теперь можно запускать:

```
cd /halflife/hlds_l/
./ cstrike_server_start
```

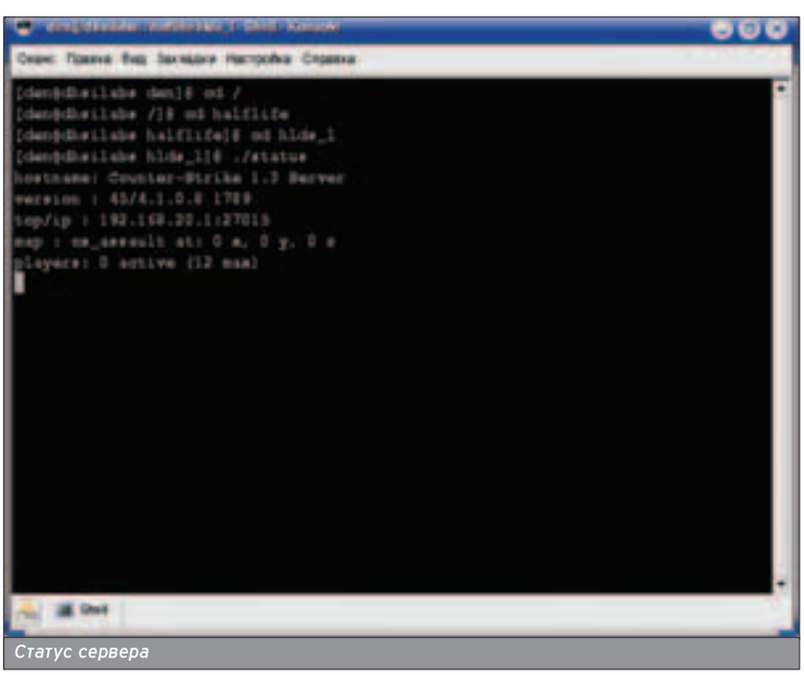
После запуска ты должен увидеть:

```
Host_Init
Added packfile /halflife/hlds_l/valve/pak0.pak (985
files)
Protocol version 45
Exe version 4.1.0.8
Exe build: 15:09:28 Sep 17 2001 (1789) WON Auth Server
couldn't exec language.cfg
Server IP address 192.168.20.1:27015
PackFile: /halflife/hlds_l/valve/pak0.pak :
models/w_battery.mdl
...
```

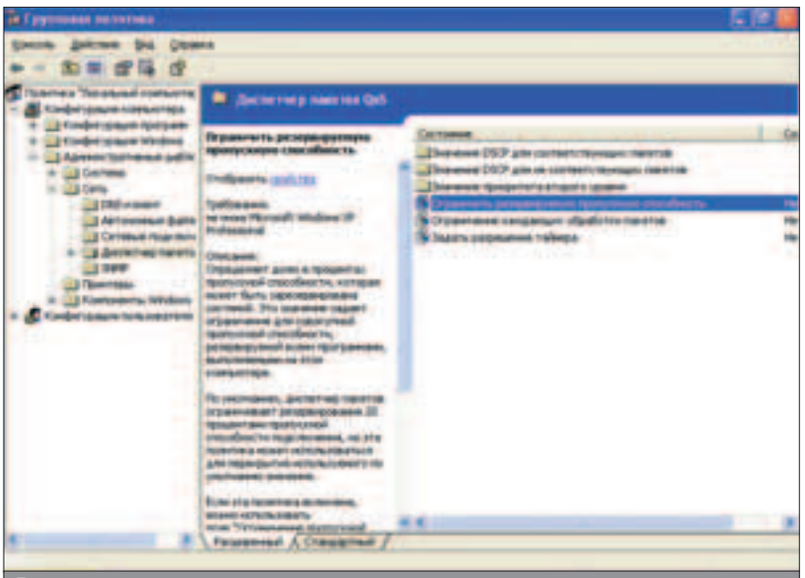
Для проверки состояния сервера используй команду status. Вот теперь сервер работает нормально. На этом настройку сервера можно считать оконченной. Если тебе понадобились дополнительные карты, помести их в каталоги /halflife/hlds_l/cstrike/ и /halflife/hlds_l/cstrike/maps/. В первый каталог помещаются wad-файлы, а во второй - bsp и txt. Наслаждайся.

ЕСЛИ У ТЕБЯ ИГРОВОЙ ЗАЛ...

Как правило, домашний игровой сервер не требует ничего особенного. Не думаю, что он будет "подтормаживать", если у тебя с десяток клиентов из твоих соседей. А вот если у тебя игровой зал, количество клиентов »



Статус сервера



Групповая политика

резко возрастает. Плюс ко всему сюда же могут подсоединяться другие игроки - из дружественных тебе игровых залов. Перед тобой будут поставлены ребром два основных вопроса:

❶. Как сделать сервер быстрее?

❷. Как защитить его?

Сделать сервер быстрее можно за счет оптимизации самого сервера Counter Strike и за счет оптимизации операционной системы Linux. Для оптимизации самого сервера тебе нужно установить патч оптимизации hlds_l_3108c_perf.tar.gz. Сначала распакуй его, а потом разберешься - там все просто :). Оптимизация Linux уже рассматривалась (см. статью в этом номере), но все же стоит рассказать об основных моментах.

Прежде всего, отключи все ненужные сервисы (команда redhat-config-services в Red Hat и drakxservices – в Mandrake). Помни главное правило: твой игровой сервер должен использоваться только как игровой сервер! Он не должен быть "по совместительству" и web-, и FTP-сервером, почтовым сервером ему тоже не стоит становиться. Все, что должно быть установлено на нем, - это только Linux, game-сервер и файрвол, например, - iptables. По возможности систему X Window тоже отключи. Для этого вовсе не обязательно удалять ее - нужно просто в файле /etc/inittab выбрать уровень запуска 3:

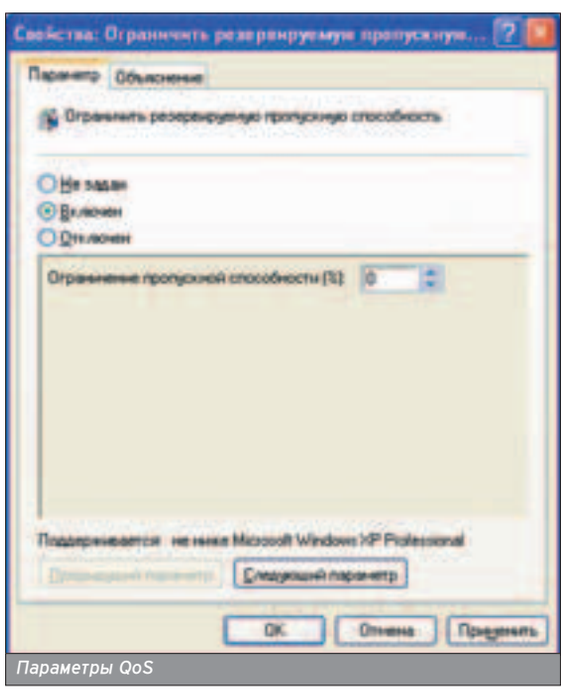
```
id:3:initdefault:
```

После того как ты отключишь ненужные сервисы, можно попытаться перекомпилировать ядро. При повторной сборке ядра обрати внимание на тип используемого процессора - установи именно твой тип процессора. Также нужно пройтись по всем опциям и отключить ненужные - чтобы не кушали ни оперативную память, ни дисковое пространство.

Следующий шаг - создание дополнительного swap-пространства. Хотя я бы порекомендовал просто купить еще 128-256 Мб оперативной памяти (всего лишь $15-30, а сервер будет работать намного быстрее). В принципе, если уж ты решил сделать именно сервер, то 256-512 Мб оперативной памяти - это необходимый минимум. После этого можно попытаться "разогнать" твой винчестер. Для этого читать man: man hdparm. В зависимости от настроек твоего дистрибутива прирост в производительности может оказаться очень ощутимым.

Чуть не забыл. Скорее всего, на машинах клиентов твоего игрового зала установлена ОС Windows XP. Не забудь отключить QoS. Для этого запусти программу gpedit.msc, перейди в раздел "Конфигурация компьютера"->"Административные шаблоны"->"Сеть"->"Диспетчер пакетов QoS". Выбери "Ограничить резервируемую пропускную способность", затем "Включен" и установи 0%. Если ты выберешь "Выключен", QoS по умолчанию будет "отъедать" от твоего канала 20%. А это очень прилично.

И о защите сервера, а точнее, как можно сделать так, чтобы никто чужой не использовал твой игровой сервер. Если ты просто установишь максимальное количество клиентов (пусть 50), то какая вероятность того, что все 50 подключенных - это "твои" клиенты? Вдруг кто-то захотел поиграть, просканировал порты и занял "место" твоего клиента, в результате чего тот не может подключиться к серверу. Конечно, можно вычислить того, кто подключается, но если ты хочешь сохранить свое время и нервы, настрой нормальный файрвол. Информации в интернете очень много. Можно использовать iptables или ipchains. Предпочтительнее, конечно, iptables - у него значительно больше возможностей. Но это уже тема совсем другой статьи.



Параметры QoS

Встретьтесь с самыми успешными российскими корпорациями:

- Внешторгбанк
- Аэрофлот - Российские Авиалинии
- Балтика
- РУСАЛ
- ТНК-ВР
- Мегафон
- Группа Северсталь
- Росгосстрах
- Kraft Foods International
- Копейка
- Бистрофф
- Московский Индустриальный Банк
- DHL Россия
- Компания "Май"
- Пивоварня Ивана Таранова
- СладКо
- BridgeTown Foods
- Лента
- Форд Россия
- РусАвтоПром
- Hines
- Wrigley Россия
- Coca Cola
- Метран
- Банк Менатеп
- Капитал Групп
- АвтоВАЗ
- Mondi Business Paper Syktyvkar
- Илим Палп Энтерпрайз
- Евросеть
- Эльдорадо
- Корпорация "Глория Джинс"
- Renault Group
- ИнвестКиноПроект

Формат конференции:

День 1: Управление бизнес-процессами в российских корпорациях

1 Марта 2005

День 2/3: Основная часть конференции

2 - 3 Марта 2005

Зарегистрируйтесь сегодня и получите эксклюзивную скидку!

Adam Smith CONFERENCES

Новое в программе 1 марта 2005

Управление бизнес-процессами в российских корпорациях

2-ая Международная конференция

Информационные Технологии в Стратегии Развития Российских Компаний

1-3 марта 2005 г., Марриотт Гранд Отель, Москва

Место встречи ведущих ИТ стратегов

Андрей Коротков
Старший Вице-президент
Внешторгбанк

Сергей Кирюшин
Генеральный директор, департамент ИТ и связи
Аэрофлот- Российские Авиалинии

Алексей Телятников
Заместитель генерального директора/Руководитель департамента ИТ
Росгосстрах

Ричард Эймс
Вице-президент по ИТ
ТНК-ВР

Михаил Эренбург
Генеральный директор департамента ИТ и организационного развития
РУСАЛ

Владимир Львов
Генеральный директор департамента ИТ
Группа Северсталь

Сергей Павлов
Заместитель генерального директора
Mondi Business Paper Syktyvkar

Игорь Ланин
Директор по ИТ
Мегафон

Геннадий Столяров
Директор по ИТ
DHL Россия

Антон Гаврин
Директор по ИТ
Компания "Май"

Сергей Робозеров
Директор по ИТ
Балтика

Сергей Гупаисов
Менеджер по информационным системам, Россия
Kraft Foods International

Спонсоры:

Монолит-Инфо · i-Teco Компания Ай-Тэко · IBS · ORACLE · i2

Медиа партнеры:

Директор · computerworld · PCWEEK · c•news · PRIME-TASS · интерфакс · Финанс. · Outsourcing-Russia.com · Большой Бизнес · The Moscow Times · Издательский дом connect · Эксперт · Коммерсантъ Деньги · Организация, поддерживающая мероприятие: АЕВ · Экспресс переводчик: TMT

НОВОЕ
Новое в программе
Управление бизнес-процессами в российских корпорациях
Вторник 1 марта 2005 года

НОВОЕ
Расширенная сессия
Создание стоимости с помощью IT
Среда 2 марта 2005 года

НОВОЕ
Заседание CIO
Анализ ключевых примеров
Четверг 3 марта 2005 года

НОВОЕ
Оптимизация отношений между поставщиком и покупателем
Четверг 3 марта 2005 года

www.russian-it.com e-mail: it@adamsmithconferences.com Тел: +44 20 7490 3774 / +7 095 775 0070 Факс: +44 20 7505 0079 / +7 095 775 0071

Колисниченко Денис dhsilabs@mail.ru

*NIX GAMES

ОБЗОР ИГР ДЛЯ LINUX

Недооценивать роль компьютерных игр на современном рынке информационных технологий нельзя. В этой небольшой статье расскажу об играх для Linux и немного об эмуляторе WineX, который позволяет запускать Windows-игры в Linux.

Что нужно, чтобы операционная система стала популярной? Эффективная в плане воздействия на потребителя маркетинговая политика (то есть то, как маркетологи представляют программный продукт - у Microsoft они работают даже лучше, чем программисты), удобство интерфейса, наличие офисных приложений, наличие приложений для работы с мультимедиа и, конечно же, наличие игр. Причем пользователю нужны нетривиальные игры: Lines и "пасьянсы" не прокатят. Современному пользователю, на столе у которого стоит "машинка", которая раньше использовалась исключительно для обработки видеоинформации, нужны максимально реалистичные игры. Linux сейчас удовлетворяет практически всем требованиям современного пользователя: она стабильна, шустра, обладает удобными графическими интерфейсами (KDE, GNOME и множество других), офисных программ и программ для работы с мультимедиа море. Остался один момент (если не считать 1С, специальных финансовых программ и САП) - игры.

LINUX-ИГРЫ

Какие же Linux-игры доступны нам сегодня?

- Heroes Of Might And Magic III (www.lokigames.com/products/heroes3)
- Freeciv (www.freeciv.org)
- Mael Storm (http://linuxgames.org.ru)
- Koules (http://linuxgames.org.ru)
- ClanBobmer (www.clanbomber.de)
- Batallion (http://evlweb.eecs.uic.edu/aej/AndyBattalion.html)
- TuxRacer (http://tuxracer.sourceforge.net)
- Ultranium III (www.jadeware.org/xeon.html)
- Quake 3 (www.quake3.com)
- Doom Legacy
- lxDoom (http://lxdoom.linuxgames.com)
- FreeCraft (www.freecraft.org)
- FlightGear

Пройдемся по этому списку и остановимся на самых популярных играх.

Heroes Of Might And Magic III

Кто же не играл в эту игру? Я не считаю себя любителем поиграть, но даже я когда-то играл в в нее, правда, под Windows. Сейчас пока доступна только demo-версия этой игры для Linux, но в скором времени, я надеюсь, выйдет и ее полная Linux-версия. Если ты заинтересовался, посети следующие странички:

❶. http://linuxgames. org.ru/get_article.php3?game=1 - здесь ты найдешь описание игры;

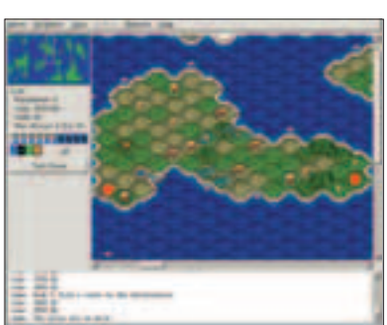
❷. www.lokigames.com/ products/heroes3/ - сайт разработчиков, здесь же можно скачать demo-версию для Linux.

Freeciv

Heroes Of Might And Magic III (www.lokigames.com/products/heroes3)

Здесь также комментарии излишни: популярная пошаговая стратегия, поддерживающая Сеть. Эта игрушка даже включена в состав некоторых дистрибутивов.

Mael Storm



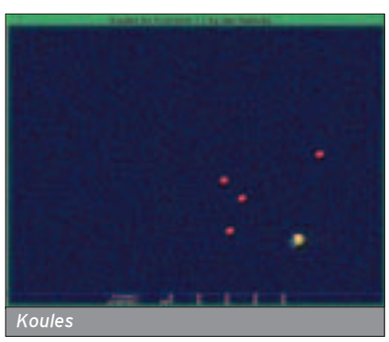
Freeciv (www.freeciv.org)

Небольшой космический симулятор. Вряд ли ты будешь играть в него долго, особенно после Windows-игр. Если интересно, на сайте http://linuxgames.org.ru можно скачать RPM с игрой.

Koules

Замечания те же, что и для Mael Storm. На сайте http://linuxgames.org.ru найдешь RPM с игрой и сможешь попробовать игрушку в действии.

ClanBobmer



Koules

Может, кто-то помнит старую игрушку Bomberman. Так вот это - ее переработанный вариант, кардинально преобразившийся, например, в плане графики и звука. Можно поиграть, чтобы успокоить нахлынувшую ностальгию. RPM-файл можно найти тут: ftp://linux.ru.net/LinuxGames/ClanBomber-1.00-1.i386.rpm

Batallion

В описании написано: "Аркада с красивой графикой". От себя честно добавлю, что ее не устанавливал. Если тебе интересно, то TAR-файл доступен по адресу ftp://autoinst.acs.uci.edu/pub/battalion/battalionLinux1.4.tar.gz.

TuxRacer

Наверное, не найдется ни одного линуксоида, который бы не играл в эту игрушку, так как она присутствует во многих дистрибутивах. Ее даже иногда называют NFS для Linux :). Это одна из самый серьезных игр для Linux: обрати внимание на графику в этой игре, и ты поймешь, о чем я говорю.

Ultranium III

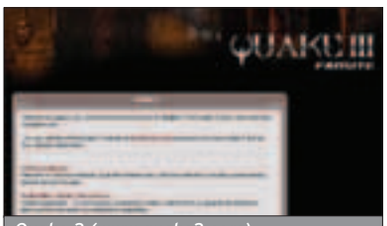
Трехмерный клон арканоида. Арканоид мне не нравится как таковой, поэтому и в Ultranium III я не играл. Описание доступно по адресу www.jadeware.org/xeon.html.

Quake 3

Выхода этой игрушки для Linux ждали очень многие линуксоиды. И действительно - отличная игрушка для отличной ОС! Правда, пока дос-

тупна только demo-версия. Мне даже показалось, что Linux-версия работает быстрее Windows-версии. Может быть, мне только показалось, потому что конкретнее сказать пока, к сожалению, не могу :(.

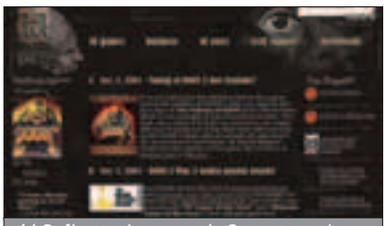
Doom Legacy, lxDoom



Quake 3 (www.quake3.com)

Старый добрый Doom, в который, наверное, играл любой, кто так или иначе имеет дело с компьютером. Теперь доступна его Linux-версия: http://3ddownloads.com/showfile.php3?file_id=39917. Другой вариант - LxDoom (Doom for Linux) - можно скачать на http://download.sourceforge.net/lxdoom/lxdoom-1.4.3.tar.gz

FreeCraft


id Software (www.quake3arena.com)

Эта свободная версия WarCraft сейчас доступна и для Linux: http://user.exit.de/johns/#download

FlightGear

Прекрасный авиасимулятор. Чтобы не отнимать хлеб у интернет-магазина LinuxCenter, ссылочку не скажу (ты и сам в состоянии найти ее), но поделюсь тем, что эта игрушка доступна на компакт-диске "Лучшие игры для Linux", который можно купить рублей за сто (www.linuxcenter.ru/?good=749).

ЭМУЛЯТОР WINEX

WineX - это эмулятор для запуска Windows-программ, которые требуют поддержки DirectX. Благодаря ему можно запускать Windows-игры под Linux. Не нужно путать WineX с Wine. WineX - совершенно отдельная разработка, которая существует в коммерческой и бесплатной версиях. За коммерческую платишь и сразу получаешь работающий эмулятор, а бесплатную нужно собирать самому из CVS. Выбор за тобой. Но я думаю, что ты справишься, а в качестве помощи могу предложить неплохой FAQ по WineX: http://unixforum.ru/index.php?showtopic=42.

Итак, давай разберемся, что можно запускать под эмулятором. В моей книге "Linux-сервер своими руками", в которой рассматривается WineX, в этом эмуляторе была протестирована работа следующих игр (это все, что было у меня на тот момент под рукой):

1. Counter Strike
2. StarCraft
3. Fallout
4. Fallout 2
5. Gunman
6. Quake 2
7. Quake 3
8. Soldier of Fortune
9. Unreal Tournament
10. Red Alert (все версии)
11. Diablo 2
12. Cesaer
13. Return to Castle Wolfenstain
14. Star track
15. Kingpin
16. Nox
17. Jadded Alliance
18. 4x4 Evolution
19. American McGee Alice
20. Daikatana
21. Heroes of Might and Magic III
22. Delta Force 1,2

Конечно, производительность практически всех продуктов ниже, чем у Windows-игр (процентов на 15-20), но играть можно. Unreal Tournament (не путать с Unreal Tournament 2004!) даже очень неплохо бегал. Небольшие тормоза с графикой компенсированы лучшей производительностью сетки (в сетевых играх). В любом случае, если тебе нужна игрушка, а ее нет под Linux и Windows ты на дух не переносишь, WineX - довольно неплохое решение.

CD: ЛУЧШИЕ ИГРЫ ДЛЯ LINUX

На этом компакт-диске, который можно купить в интернет-магазине LinuxCenter, ты найдешь:

- demo-версию игры Unreal Tournament 2004 Demo;
- demo -версию игры UFO for Linux;
- Flight Gear - мощный авиасимулятор.

Кроме того, на этом CD есть инсталляторы игр Max Payne 1 и 2, Quake 2 и 3, Soldier of Fortune 2 для WineX. Тебе ничего не придется настраивать, а только нажать кнопку Next и найти затерявшийся компакт-диск с Windows-версией игры.

W W W

- Russian Linux Games Site - **http://linuxgames.org.ru**
- Новости мира Linux-игр - **www.tuxgames.ru**
- Свободные игры - **http://zavar.narod.ru/games.htm**

Roman AKA Docent (dOcent@rambler.ru), Колисниченко Денис dhsilabs@mail.ru

ЛУЧШИЙ СОФТ ПОД *NIX

ОБЗОР ПОЛЕЗНОГО ПО ПОД *NIX-СИСТЕМЫ

Под *nix существет огромное количество самого разного софта. Кроме того, почти все необходимые программы поставляются уже вместе с дистрибутивом. В некоторых дистрибутивах встроен еще и мощный набор офисных программ Open Office, не уступающий по возможностям MS Office. Попробуй представить себе, что в Windows был бы встроен сразу и Office, и клиент ICQ, и WinAmp, и еще много всего другого. А в дистрибутиве *nix это все еще и на выбор. Хочешь - ставь такой плейер. Не нравится - вот тебе несколько других на выбор. Не хочешь этот текстовый редактор - выбирай другой: на любой вкус найдется. Так что поговорим сегодня о разных полезных в хозяйстве программах. Многие из них можно найти в дистрибутивах *nix, а можно скачать с сайтов разработчика.

XMMS

Сайт: http://xmms.org
Размер: 1,9/3 Мб (tar.gz/rpm)
Текущая стабильная версия: 1.2.10

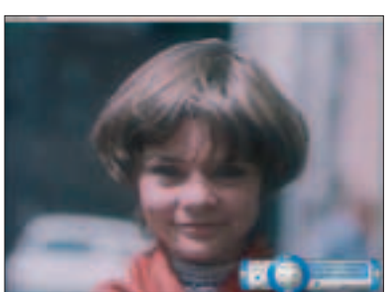
XMMS - лучший плейер для Linux.

Без музыки, как известно, никуда. Как говорится в первоисточнике, нам песня строить и жить помогает. Поэтому мы и начнем с обзора проигрывателя музыки для пингвина. По этой же причине начну обзор с лучшего на сегодняшний день аудиопроигрывателя для *nix - XMMS. Внешне он брат-близнец WinAmp в Windows, и, кстати, он даже поддерживает его шкурки и playlist'ы. Кроме того, этим проигрывателем давно укомплектовывается практически любой дистрибутив Linux. Плейер умеет проигрывать MP3, .wav, .mod, audio-CD и многое другое. Для него существуют различные плагины и расширения. К твоим услугам эквалайзер с настройкой различных эффектов звучания, редактор playlist'а и экранные заставки. В общем, аналог WinAmp.

XMMS - лучший плейер под Linux

MEDIA PLAYER

Сайт: www.mplayerhq.hu
Размер: 2,1 Мб
Текущая стабильная версия: 0.93

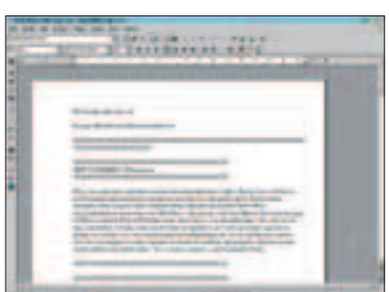

Media Player (на экране фильм "Гостья из будущего" в MPEG4)

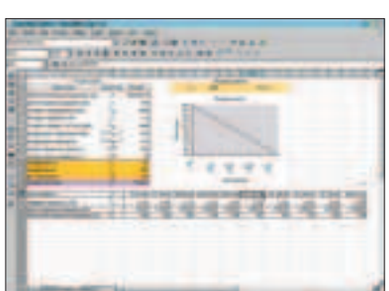
Следом за музыкой идут и фильмы. Кино тоже нравится нам всем, и, конечно, хотелось бы смотреть его в *nix на хорошем, шустром плейере, который поддерживает множество кодеков. Есть такой плейер - Media Player (MPlayer). Довольно компактен и без проблем читает MPEG4, DivX и их производные. Кроме этого, можно скачивать дополнительные кодеки с сайта родного и стороннего разработчика. Таким образом, плейер может проигрывать почти все существующие видеоформаты, а в нагрузку и музыкальные. Из приятных мелочей - стандартная для многих программ *nix поддержка скинов и шрифтов.

OPENOFFICE

Сайт: http://openoffice.org
Размер: 130-205 Мб (rpm/gz)
Текущая стабильная версия: 1.1.3

Полноценный офисный пакет, по возможностям не уступающий, а можжет, даже превосходящий MS Office. В него входит набор полезных для офиса и дома приложений, среди которых особо примечательны текстовый редактор, электронные таблицы и редактор векторной графики и презентаций. Все эти программы понимают


Текстовый редактор OpenOffice - M$ Word отдыхает!


Электронные таблицы OpenOffice. Теперь в твоем офисе нет места для Windows

множество форматов и могут экспортировать их, например, в .doc и .xls для разных версий MS Office. Редактор векторной графики может работать с известными форматами Corel Draw (.cdr), Adobe Illustrator (.ai) и AutoCAD (.dxf). Не может не радовать и существование русскоязычного пакета OpenOffice. И все это удовольствие, заметь, совершенно бесплат-

ное, а некоторые дистрибутивы Linux уже имеют этот пакет в своем составе. А что еще может быть нужно для плодотворной работы?

FIREFOX

Сайт: www.mozilla.org/products/firefox/index.html
Размер: 8,9 Мб
Текущая стабильная версия: 1.0

Firefox - браузер и ничего лишнего

Браузер и ничего кроме браузера. Firefox очень похож на Mozilla, облегченной версией которой по сути и является. Очень радует отсутствие лишних кнопочек, фенечек и баннеров - все только самое нужное для просмотра страниц в интернете. Браузер поддерживает все современные интернет-технологии, такие как Java, JavaScript и Flash, умеет блокировать всплывающие окна, имеет встроенную панель поиска Google. И еще одна полезная фишка этого браузера - он умеет экспортировать настройки, закладки и cookie из других браузеров, так что переход на Firefox с Mozilla или Opera (а Windows- версия экспортирует все и из IE) будет совершенно безболезненным.

WGET

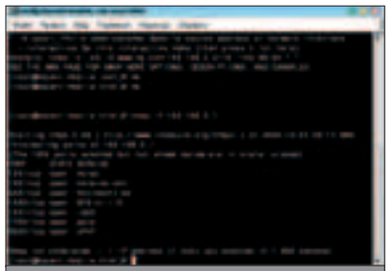
Сайт: http://wget.sunsite.dk
Размер: 1,3 Мб
Текущая стабильная версия: 1.9.1

WGet - интернет-пылесос

Мощная утилита для скачивания файлов по HTTP- и FTP-протоколу. Выкачивает файлы даже по плохой телефонной линии с невысокой скоростью скачивания. Позволяет докачивать файлы и использует многопоточную загрузку. Единственный минус, наверное, в отсутствии графического интерфейса: общаться с программой можно только через командную строку. Но нам, линуксоидам, к этому, в общем, не привыкать.

NMAP

Сайт: www.insecure.org/nmap
Размер: 30 Кб
Текущая стабильная версия: 3.75



Знаменитый сканер и анализатор Nmap

Мы уже как-то упоминали в нашем журнале Nmap. Это многофункциональный сканер портов и сетевой анализатор, который засветился даже в фильме "Матрица". Программа входит в состав Linux, но, тем не менее, с сайта разработчика всегда можно скачать более свежую версию и различные примочки к ней, например, графический и web-based интерфейсы. Эта программа покажет тебе, какие порты открыты на проверяемой тобой системе, что за сервисы там установлены, какая операционная система используется и какой она версии. Маленькая шустрая утилита, без которой не обойдется ни администратор, ни хакер.

XCHAT

Сайт: http://xchat.org

XChat - отличный IRC-клиент

Превосходный IRC-клиент, использующий библиотеку GTk2+. До появления ICQ IRC-клиенты были основными программами "для человеческого общения". Однако даже сейчас их продолжают использовать - кто-то по привычке, а кому-то IRC нравится больше, чем ICQ. Программа Xchat входит в состав большинства дистрибутивов Linux и в большинстве случаев устанавливается по умолчанию. Ее не нужно откуда-то загружать и устанавливать - она всегда под рукой...

licq - ICQ для Linux

LICQ

Сайт: http://licq.org
Текущая стабильная версия: 1.3.0

Что такое ICQ, знают все. А вот о том, что есть версия ICQ для Linux (licq), некоторые пользователи даже и не догадываются. Как и Xchat, присутствует в большинстве дистрибутивов и устанавливается по умолчанию. Проста в использовании и похожа на обычную ICQ для Windows.

psi

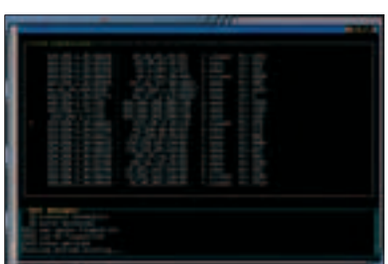
PSI

Сайт: http://psi.affinix.com
Текущая стабильная версия: 0.9.3

Клиент сети для быстрого обмена сообщениями Jabber. Может не входить в состав некоторых дистрибутивов. Для ее загрузки посети сайт разработчиков: http://psi.affinix.com. Если у тебя ALT Linux, то программа уже будет установлена. Если это не так, установи пакет psi-0.9.2-alt2.src.rpm

ETTERCAP

Сайт: http://ettercap.sourceforge.net



ettercap - и ты знаешь о трафике все

Текущая стабильная версия: 0.7.2

Ettercap - это утилита для анализа сетевого трафика. Что о ней говорить - на то он и анализ, чтобы запустить его и узнать все самому. Опять же, программа может не входить в состав твоего дистрибутива. Если это так, можно попытаться загрузить ее с сайта ALT Linux или по адресу http://ettercap.sourceforge.net.

KMAIL

Сайт: http://kmail.kde.org

Основной почтовый клиент KDE. Я бы даже сказал, что это основной почтовый клиент Linux. На самом деле это очень удобная и функциональная программа. Нужно отметить, что программа появилась давненько и постоянно обновляется, то есть является востребованной.

kmail для чтения почты в KDE

Валуев Владимир CW (cw@chatnet.ru)

ЛИЧНАЯ IRC-СЕТЬ

УСТАНОВКА И НАСТРОЙКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ IRC

История создания ircd началась в далеком августе 1988 года (протокол IRC, таким образом, старше http). С тех пор появилась масса разнообразных IRCd и выбор программного обеспечения IRC перестал быть таким узким, каким был когда-то. В этом материале я постараюсь помочь тебе, начинающему IRC-оператору, с выбором, установкой и настройкой своего собственного персонального irc-сервера. Разве ты никогда не мечтал оказаться по ту сторону баррикад? (я мечтал, особенно когда k-line'ом вылетал с далнета, - прим. Горл)

Прежде чем устанавливать ircd, определись с выбором софта. К твоим услугам множество различных по уровню функциональности и стабильности ircd. Наиболее известные из них:

IRCD-Hybrid (http://ircd-hybrid.com). Старейший и наиболее известный ircd. Именно этот демон используется на большинстве серверов сети EFnet. Hybrid отличается стабильностью работы и гибкостью настройки. Поддерживает IPV6, сжатие трафика и OpenSSL (только для серверов).

Bahamut (http://bahamut.dal.net). Стандарт де-факто на серверах сети DALnet. Был основан на DreamForge и IRCD-Hybrid. Содержит немало функций для организации качественной IRC-сети, но мало развивается в отличие от других ircd, таких как UnrealIRCD или IRCD-Hybrid.

UnrealIRCd (www.unrealircd.com). Возможности сервера очень велики. Тут и поддержка IPV6, и SVS для сервисов, и OpenSSL, как для серверов, так и для клиентских подключений (очень полезная для клиента вещь, особенно когда он работает из локальной сети). Unreal работает со множеством сервисов, команды его неисчислимы, но и у него есть один "незначительный" минус - плохо держатся соединения между серверами. Как следствие - частые split'ы.

Не стоит забывать и о серверах, в разработке которых принимали участие наши соотечественники. Но, как я уже заметил выше, многие русские проекты представляют собой сборища патчей и обновлений известных движков.

ircd-RU (www.ircd.ru). Bahamut с добавленной в него поддержкой кириллицы и других кодировок в названиях ников и каналов. Активно используется в сети WeNet.

IRCD-Hybrid-RU (http://hyb.net.ru). Адаптация IRCD-Hybrid под нужды российских пользователей. Теперь в демоне можно обнаружить поддержку русских символов в никах и названиях каналов, и SSL для клиентов, и SVS-команды для сервисов. В общем, возможностей очень много, всех не перечислишь.

УСТАНОВКА IRCD

Большинство используемых на данный момент IRC-серверов работают на "Гибриде" и его производных, поэтому было бы разумно рассмотреть установку именно на примере этого ircd. Установки всех модификаций мало отличаются друг от друга. Для русского контингента лучшим выбором, по моему мнению, станет IRCD-Hybrid-RU. На примере именно этого дистрибутива покажу и расскажу, как устанавливать и настраивать.

Для начала скачай последнюю версию дистрибутива с сайта разработчика и разархивируй ее в какое-нибудь удобное место.

Теперь заходи в директорию, в которую ты распаковал дистрибутив, и запускай ./configure. Этот сценарий настраивает все необходимое, самостоятельно отыскивает нужные библиотеки и т.п. По умолчанию ircd устанавливается в каталог /usr/local/ircd. Если у тебя есть желание изменить этот путь, добавь ключ "--prefix=/path/to/install".

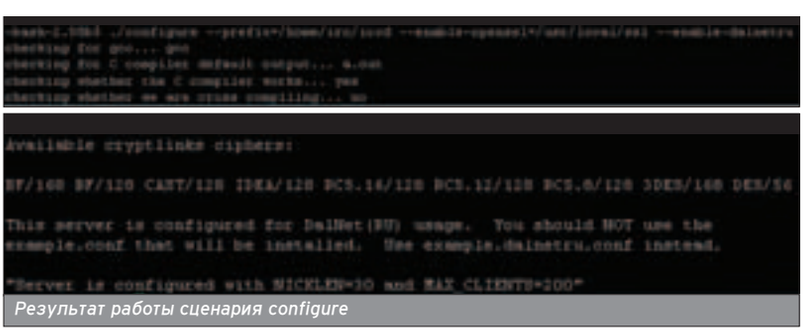
Вероятно, тебе захочется включить SSL. На самом деле, конфигуратор должен сам найти пакет OpenSSL автоматически, но если вдруг он этого не сделает - от тебя потребуется добавить ключ "--enable-openssl=/path/to/openssl", в котором указать полный путь к директории, в которой установлен.

Интересных ключей очень много, увидеть весь список ты можешь добавив ключ "--help" к конфигуратору. Вот только часть тех ключей, которые могут предстать перед твоим взглядом:

--disable-assert - запрещение отладки (может пригодиться, если конфигу-

> Большинство используемых на данный момент IRC-серверов работают на "Гибриде" и его производных.

Загрузка дистрибутива



Результат работы сценария configure

Сборка ircd

Для любителей покрасоваться хостом есть блок auth, в котором можно прописать себе любой хост.

ратор по тем или иным причинам не работает);

--with-nicklen – задает длину ника;

--with-topiclen – задает длину топика на канале;

--with-maxclients - задает максимальное количество соединений (по умолчанию в IRCD-Hybrid-RU – 200).

Если ты не уверен в выборе каких-либо опций, можешь задать единственный ключ --enable-dalnetru, при помощи которого все основные параметры ircd будут настроены автоматически.

Сконфигурировав ircd, ты имеешь полное право его, наконец, установить. Собирай проект командой make.

Если все прошло без ошибок, то запускай процесс установки make install.

После инсталляции нужно зайти в директорию, куда ты установил ircd и создать пустые файлы для логов:

```
$ cd /path/to/install
$ touch logs/userlog
$ touch logs/operlog
$ touch logs/foperlog
```

Также не забудь скопировать папку doc/codepage (файлы кодировок) в директорию /etc установленного ircd.

Разобравшись с установкой сервера, можно приступать к настройке.

НАСТРОЙКА IRCD

Под настройкой ircd подразумевается настройка файла ircd.conf.

В этом файле ты указываешь имя своего сервера (M-line); прописываешь себя администратором (A-line); настраиваешь параметры соединений (Y-lines), прописываешь себе статус оператора (O-lines), при необходимости закрываешь вход на сервер паролем (I-line); устанавливаешь C-lines, если намерен куда-либо линковаться; U-lines - для того чтобы твой сервер "слушался" сервисов.

Давай рассмотрим каждый пункт более подробно.

M-line:

```
serverinfo {
  name = "server.dalnet.ru"; - название сервера
  description = "Test Server Dalnet.ru, MSK, Russia"; - описание сервера
  network_name = "DALNet.RU"; - название сети
  network_desc = "Russian IRC Network"; - ее описание
  max_clients = 512; - максимальное количество пользователей, разрешенное на твоем сервере  hub = yes;
 # rsa_private_key_file = "/usr/local/ircd/etc/rsa.key";
 # ssl_certificate_file = "/usr/local/ircd/etc/cert.pem";
 };
```

A-line:

```
admin {
  name = "Smurf target";
  description = "Main Server Administrator";
  email = "<syn@packets.r.us>";
 };
```

Параметры соединений (Y-lines) по умолчанию уже настроены. В случае линка в какую-нибудь irc-сеть настрой их согласно требованию линк-администратора.

Для любителей покрасоваться хостом есть блок auth, в котором можно прописать себе любой хост.

```
auth {
  user = "*you.ident@your.ip";
  spoof_notice = no;
  exceed_limit = yes; - снятие ограничения на количество коннектов с одного ip
  kline_exempt = yes; - защита от kline
  gline_exempt = yes; - хост будет защищен от gline (кроме сервисного)
  can_flood = yes;
  have_ident = no;
  class = "users";
  no_tilde = yes; - убирает "~" перед ident'ом
  spoof = "your.host";
 };
```

Кроме всего этого, в блоке auth ты можешь настроить для любого класса соединения и/или зоны свои привилегии.

С блоком Operator (O-lines) все просто. Смело вписывай в name свой ник, в user вставляй свой *ident@host, выставляй себе необходимые привилегии и генерируй операторский пароль при помощи утилиты mkpasswd в директории bin установленного ircd.

В блоке channel находятся опции, относящиеся к настройкам каналов. Тут будь внимателен с опциями max_chans_per_user и max_bans. Дефолтовые значения этих параметров, как правило, не устраивают администраторов. Также не рекомендуется ставить запредельные значения вроде 100 или 200, так как ircd в этом случае будет потреблять немало ресурсов сервера, на котором он установлен.

Блок serverhide рекомендую оставить без изменений. При линке в какую-либо irc-сеть он настраивается согласно принятым требования безопасности в данной сети.

Блок general содержит основные настройки ircd, которые можно задать как в самом конфигурационном файле, так и в config.h перед сборкой ircd. Настройки по умолчания лучше не трогать.

Для того чтобы твой ircd "слушался" сервисов, существует блок shared (U-lines):

```
shared {
  name = "services.dalnet.ru";
 };
```

Линк на сервисы или другие сервера прописывай в блоке connect:

```
connect {
  name = "services.dalnet.ru"; - название сервера
  host = "255.255.255.255"; - его ip или host
  send_password = "servicespasswd"; - пароль "туда"
  accept_password = "servicespasswd"; - пароль "оттуда"
```

»

IRCD-Hybrid-RU - адаптация IRCD-Hybrid под нужды российских пользователей.

В демоне можно обнаружить и поддержку русских символов в никах и названиях каналов, и SSL для клиентов, и SVS-команды для сервисов.

Сеть DALnet основана на Bahamut.

После инсталляции нужно зайти в директорию, куда ты установил ircd, и создать пустые файлы для логов.

```
-bash-2.05b# make install
build ==> modules
make[1]: Entering directory `/home/irc/ircd-hybrid-ru/modules'
make[1]: Nothing to be done for `build'.
make[1]: Leaving directory `/home/irc/ircd-hybrid-ru/modules'
build ==> adns
make[1]: Entering directory `/home/irc/ircd-hybrid-ru/adns'
```

Установка ircd

```
-bash-2.05b# cd ircd/bin
-bash-2.05b# ./ircd
ircd: version hybrid-ru-7.0rc10
ircd: pid 3113
ircd: running in background mode from /home/irc/ircd
```

Запуск ircd

```
Beginning Services configuration.

Note: press Return for the default, or enter a new value.
In what directory do you want the binaries to be installed?
[/home/irc/services]
/home/irc/services does not exist.  Create it?
[y]

Where do you want the data files to be installed?
[/home/irc/services] /home/irc/services/data/
/home/irc/services/data/ does not exist.  Create it?
[y]
```

Загрузка дистрибутива

При наличии шифрования будут недоступны такие функции, как GETPASS.

```
    port = 6668; - удаленный порт, на который осуществляется линковка
    hub_mask = "*"; - маска, необходимая для соединения хабов и сервисов (для корректной работы JUPE)
    class = "server"; - класс соединения, для хабов рекомендуется разделять серверный класс на uplink и downlink
    autoconn = no;
    compressed = no; - сжатие трафика
    cryptlink = no; - шифрование данных
  };
```

Теперь осталось отредактировать пути к логам сервера, файлам кодировок и модулям, и можно будет запускать ircd, для чего есть волшебная команда ./path/to/install/bin/ircd.

Если ты все сделаешь правильно, то картина на твоем экране будет напоминать то, что ты видишь на скриншоте.

УСТАНОВКА SERVICES

Сервисы (services) позволяют пользователям irc-сети регистрировать свои ники, каналы, управлять каналами, обмениваться короткими сообщениями и т.п., то есть обладают функциональностью, которой не хватает ircd. Наиболее популярными сервисами являются:

IRC Services (www.ircservices.za.net) - хорошие сервисы с неплохим набором функций. Созданы на модульной основе.

Anope (www.anope.org) - эти сервисы открывают перед тобой самые широкие возможности, поддерживают множество ircd, есть поддержка модулей и русский help. Основаны на IRC Services. Существуют версии под Linux/Win32.

HybServ (http://kreator.esa.fer.hr) - сервисы для IRC-серверов на основе IRCD-Hybrid.

Auspice (http://sourceforge.net/projects/auspice) - сервисы опять же с колоссальными возможностями. Множество сервисных ботов, крайне нестабильные сервисы. Если устанавливать, то только в целях ознакомления или удовлетворения любопытства. Существуют версии под Linux/Win32.

Anope - в настоящее время единственные сервисы, в которых есть поддержка IRCD-Hybrid. HybServ практически не обновляется разработчиками, в то время как Anope, на примере которого расскажу и покажу установку, делает это ежемесячно.

Скачивай дистрибутив с сайта разработчика и разворачивай его. В директории, куда разархивировал, запускай ./Config - это сценарий для настройки перед установкой. Здесь от тебя потребуют ответы на несколько вопросов. Если тебя устраивают варианты ответов по умолчанию - жми <ENTER> или указывай свой вариант. Однако единственный вопрос, на который ты должен дать ответ самостоятельно, - это выбор ircd. В нашем случае это Hybrid-IRCd 7.0.

Не рекомендую использовать шифрование паролей в базах данных. При наличии шифрования будут недоступны такие функции, как GETPASS, и со временем ты поймешь, что это крайне неудобно. Перейти "на лету" на вариант без шифрования не получится или получится с полной потерей базы данных, так что об этом стоит подумать заранее.

Далее для сборки сервисов набирай (g)make.

Для установки в указанную директорию (g)make install.

На этом процесс установки прошу считать оконченным. Наступает пора переходить к настройке сервисов - последней настройке на сегодня.

Anope - отличные сервисы. Предоставляют самые широкие возможности, поддерживают большое количество ircd, есть поддержка модулей. Основаны на IRC Services. Есть русский help. Существуют версии под Linux/Win32.

По ходу конфигурации сервисов нужно будет указать, каким ircd ты пользуешься. В этом материале рассматривается: Hybrid-IRCd 7.0

```
Select the closest to the type of server on your IRC network:
    1) DreamForge 4.6.7     [dated IRCd, upgrade to a current one]
    2) Bahamut 1.4.27       [or later]
    3) UnrealIRCd 3.1.1     [or later (not 3.2)]
    4) UltimateIRCd 2.0.2   [or later]
    5) UltimateIRCd 3.0.0   [alpha26 or later]
    6) HyDrid IRCd 7.0      [experimental]
    7) ViagraIRCd 1.3.x     [or later]
    8) PTlink 6.15.0        [experimental]
    9) RageIRCd 2.0.0       [beta-6 or later]
   10) Unreal 3.2           [Unreal 3.2 beta19 or later]
[no default] 6
```

```
Do you want to use the MD5 message-digest algorithm to encrypt passwords?
(Selecting "yes" protects your passwords from being stolen if someone
gains access to the Services databases, but makes it impossible to recover
forgotten passwords. There is no way to reverse this operation, so make
sure you really want to enable it.)
[no]

Allow anope to automatically check for mysql libraries?
Unless you get errors with make, there is no need to
change this setting.
[yes]

Saving configuration results in config.cache... done.
./configure  --with-bindir=/home/irc/services  --with-datadir=/home/irc/services/data/  --with-ircd=IRC_HYBRID  --with-pers
istence=OTF
checking for gcc... gcc
```

Сборка сервисов

Сборка сервисов

Установка сервисов

Те серверы, названия которых отличаются от этой маски, GLOBAL не увидит.

бы в случае падения одного сервера сеть не осталась без сервисов.

Подобным образом настраиваются имена сервисов:

ServerName "services.dalnet.ru"

Описание сервисов в /whois или /info:

ServerDesc "Services for DALNet.RU IRC Network"

Маска сервисов:

ServiceUser services@dalnet.ru

Теперь выбирай сами сервисы (nickserv, chanserv,..,operserv), которые ты хочешь видеть в сети. Один нюанс: Anope поддерживают различные стандарты ircd. В этом можно было убедиться тогда, когда мы собирали сервисы и выбирали тип ircd, но не все ircd поддерживают существующие в Anope сервисы. В нашем случае в Hybrid IRCd нет поддержки HostServ, поэтому эту строчку стоит закомментировать.

Блок Services data filenames оставляй без изменений и переходи к блоку Network information. Здесь тебя должны заинтересовать следующие опции:

NetworkDomain "dalnet.ru" – домен сети. Тут следует отметить, что команда GLOBAL будет работать только в том случае, если сервер(ы) оканчиваются на *.dalnet.ru. Те серверы, названия которых отличаются от этой маски, GLOBAL не увидит. Исправить положение можно перечислив все домены сети через пробел, например, "dalnet.ru chatnet.ru"

NetworkName "DALNet.RU" – название сети

Разобравшись с этими опциями, можно переходить к базовым настройкам сервисов:

UserKey1|2|3 – обязательный параметр. Ты обязан раскомментировать и изменить дефолтовые значения на любые собственные.

UserKey1 4567978

UserKey2 2398546

UserKey3 8763456

Остальные параметры здесь можно оставить без изменений.

Для работы SENDPASS настраиваешь блок Mail-related options, а именно: указываешь путь к sendmail и e-mail, от которого будут отсылаться сообщения.

Далее идет настройка каждого сервиса NickServ, ChanServ, MemoServ... Рассмотрю самые важные опции для каждого сервиса:

Настройки NickServ:

NSDefLanguage – устанавливает язык сервисов по умолчанию. Ставь по умолчанию русский язык [11].

NSExpire - время, по прошествии которого истекает регистрация ника, если его не идентифицировать.

NSMaxAliases – максимальное количество ников для группы. Установка 0 снимает любые ограничения.

NSAccessMax – максимальное число записей в листе доступа к нику.

NSMaxAliases – включение этой опции предотвращает использование администраторами сервисов команд DROP, FORBID, GETPASS, SET PASSWORD в отношении других сервис-администраторов.

Для работы SENDPASS настраиваешь блок Mail-related options.

Настройки ChanServ:

CSMaxReg - максимальное количество каналов, которое можно зарегистрировать на один ник.

CSExpire – время, по прошествии которого истекает регистрация канала, если его не идентифицировать.

CSAutokickMax – максимально возможное число записей в autokick list. Значения по умолчанию, как правило, не хватает :).

Настройки MemoServ:

MSMaxMemos – максимальное количество memo-сообщений, которое может храниться у одного пользователя.

MSSendDelay – промежуток (в секундах) между отправкой memo-сообщений.

Настройки OperServ:

ServicesRoot - суперпользователь сервисов. По умолчанию параметр закомментирован. Ты просто обязан раскомментировать его и вставить свой ник. Если понадобится, можно указать несколько ников через пробел, например: ServicesRoot "CW Megamozg" SuperAdmin – дает права фаундера всех каналов и обеспечивает возможность изменять mode пользователей, если это поддерживает протокол ircd.

Включением/выключением параметров WallOs[Global], [Mode], [Clearmodes], [Akill]... ты можешь регулировать появление WALLOPS/GLOBOPS от сервисов на соответствующие действия. Во избежание лишнего флуда в окне Status'a большую часть опций можно закомментировать.

На этом мое описание базовых настроек сервисов завершается. За более тонкой настройкой (DefCon, MySQL) желающие могут обратиться к докам.

Переименовывай example.conf в services.conf и запускай сервисы командой ./services.

Если все OK, то ты обнаружишь поразительное сходство Status'a IRC-клиента со скриншотом.

Теперь регистрируй ник, который ты указал в ServicesRoot, идентифицируй его и... ты Services root!

link established ;)

Content:

Косякин Антон (deil@real.xakep.ru)

С В *NIX - ЗАЛОГ ЗДОРОВЬЯ

ОСНОВЫ ПРОГРАММИРОВАНИЯ В *NIX-СИСТЕМАХ

Программирование - это в первую очередь свобода. Обладая теми или иными программистскими навыками, ты получаешь свободу действий, которой так не хватает в жизни. Свобода же - это комфорт. А как можно пользоваться операционной системой, если это некомфортно? В общем, я хотел сказать, что без программирования никуда, особенно в *nix. Будем учиться.

Многим известно, что UNIX (*nix) - универсальная среда программирования. ОС, созданная программистами для программистов. Здесь используется множество всевозможных языков программирования (да-да, знаю: это множество даже не более чем счетно и очень даже конечно), один из которых ты можешь выбрать для решения поставленных перед тобой задач. Perl, Python, Java, Tcl/tk, UNIX Shell Script, C#, Pascal, BASIC, C/C++... Так сложилось (и надеюсь, так будет еще долго), что основным языком программирования, который используется в семействе *nix-систем, является C. Поэтому речь пойдет именно о нем.

Сразу хочу заметить - эта статья не о самой ОС. И не о C. А о программировании на этом языке под эту ОС. Основные навыки работы в *nix-системе, умение программировать на языке C, знание того, что такое компилятор :), умение пользоваться man'ами (для этого конкретного случая наиболее интересными являются разделы 2 и 3, охватывающие системные вызовы и функции стандартных библиотек) - и можно приступать.

IDE

Для начала нужно определиться, где набивать исходные тексты своих будущих программ. Можно пользоваться либо одним из обыкновенных и многочисленных текстовых редакторов, либо остановить свой выбор на какой-нибудь из IDE, которых, правда, чуточку меньше :).

Из сред разработки под *nix могу предложить таких монстров, как, например, KDevelop, Anjuta. Также можно попробовать что-нибудь консольное - Rhide или Motor. Если их нет в дистрибутиве, то положение исправит freshmeat.net с его возможностью поиска. Об упомянутых выше средах разработки могу сказать лишь то, что это самые обычные IDE со стандартным набором функций, в которых легко разобраться. В случае чего - поможет встроенная справка. Так что выбирай то, что больше нравится. Однако я придерживаюсь мнения, что самая удобная и полная IDE для ОС *nix - сама ОС *nix. Поэтому я уже давно пользуюсь обычным текстовым редактором, компилирую из командной строки, а при отладке использую gdb.

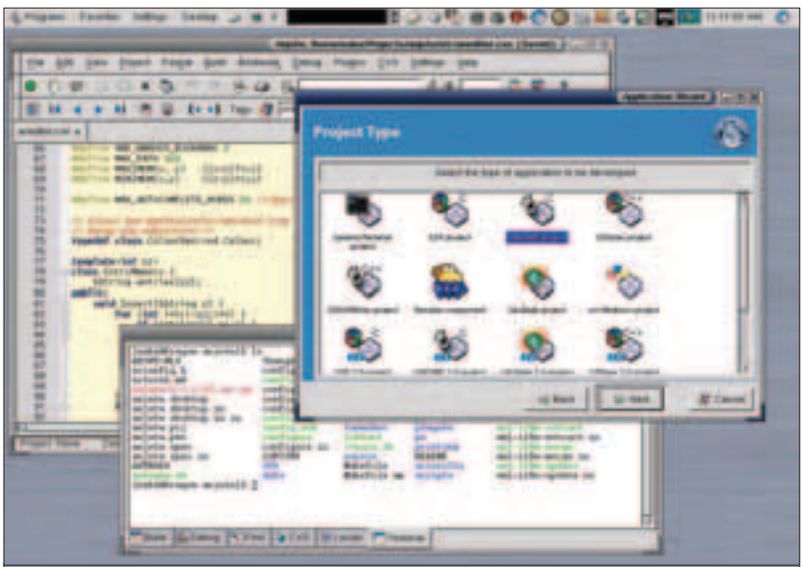
СОБИРАЕМ

Следующим в нашем списке стоит компилятор. В Linux, как и в GNU-системе, выбор очевиден - это GCC. Можно, конечно, найти какую-нибудь альтернативу, например, icc - Intel C Compiler, но это вряд ли тебе нужно.

Набрал код своей программы и сохранил его в некотором файле - следующим этапом будет компиляция. В случае использования GCC для этого необходимо просто в командной строке набрать "# gcc program.c". На выходе получишь исполняемый файл a.out или увидишь список ошибок, которые препятствуют компиляции. Для изменения названия выходного файла используется параметр "-o": # gcc -o program program.c. В таком случае у нас вместо a.out появится файл Program или снова покажется список ошибок :).

Довольно часто получается так, что исходный код программы разрастается до таких размеров, что держать его в одном файле становится нецелесообразно, и тогда его (исходник) разбивают на модули. В таком случае при компиляции такой программы понадобится ключ "-c" (# gcc -c program1.c): GCC выдаст объектный файл с расширением .o, который будет использоваться в дальнейшем. Скомпилировав таким образом все модули нашей программы, собрать их в один исполняемый файл можно такой вот командой: # gcc -o program program1.o program2.o ... .

Когда написание программы в принципе завершено и когда пора отправлять ее на растерзание конечным пользователям, можно использовать оптимизацию для уменьшения размера или ускорения работы программы. Параметры оптимизации задаются ключами "-O<x>". Использование "-O0" отключает оптимизацию. Используется по умолчанию. При использовании O1 компилятор выполняет набор некоторых оптимизаций, влияющих на скорость выполнения и уменьшение размера выходного файла. Однако применяются только те типы оптимизаций, которые не увеличивают время компиляции. "-O2" включает почти все типы оптимизаций, увеличивая при этом время компиляции и производительность выходного кода. Однако развертывание циклов и выполнение оптимизации подстановок (inline'инг) не происходит. Далее следует ключ "-O3", который включает оставшиеся виды оптимизаций, такие как упомянутый выше inline'инг. Еще следует упомянуть ключ "-Os", который



IDE Anjuta в действии

включает все виды оптимизаций, включающихся ключом "O2", которые не увеличивают размер выходного кода. Плюс еще кое-что. За более подробной информацией обращайся к info-страницам по GCC.

Для отладки написанной программы необходимо совершить еще одно телодвижение - добавить отладочную информацию к программе. Для этого при компиляции (с использованием GCC) необходимо добавить ключ "-g". Именно он и сделает то, что нам нужно. Вообще, отладочная информация может быть представлена в нескольких форматах. Один из них - формат, "разработанный" специально для использования с gdb. Для осуществления этого есть специальный ключ - "-ggdb".

Одной интересной особенностью компилятора GCC является тот факт, что ключи g и O<x> могут быть использованы одновременно. Так можно попросить компилятор оптимизировать код, а затем добавить отладочную информацию. И в итоге получаешь результаты такой оптимизации в наглядном виде. Некоторые константы и переменные могут исчезнуть, блоки кода - изменить свое положение в программе. А некоторые - вообще не исполняться, так как их результат заранее можно просчитать и он никогда не меняется. Кстати, можно задать количество отладочной информации, помещаемой в результирующий файл. Всего существует три уровня, второй используется по умолчанию. Задается все ключом "-g<x>" или "-ggdb<x>". Уровень один - минимум, уровень три - максимум (добавляется информация о макросах в программе). Такие вот дела.

НАЧИНАЕМ

Итак, теперь перейдем к главному: программированию на C. По-моему, здесь все поделено на две части: использование системных вызовов ядра ОС, предоставляющих некоторые основные функции, и использование сторонних (внешних) библиотек, которые предоставляют более удобный интерфейс для некоторых возможностей ОС. Поэтому все сводится к чтению описаний системных вызовов и гайдов по использованию внешних библиотек :).

Если твои программы соответствуют стандарту ANSI C или C99, то они будут легко компилироваться под любой ОС с использованием "правильного" компилятора. Однако стремление к соответствию упомянутым стандартам, к сожалению, не разбудит полностью все таланты системы.

Для любой *nix-программы, помимо стандартных заголовков stdlib.h & stdio.h, необходимо использование заголовка unistd.h. При использовании некоторых специфичных типов данных необходимо подключать заголовок sys/types.h.

Для примера рассмотрим основные функции (примитивы) для работы с файлами, предоставляемые любой *unix-системой. Эти функции состоят из небольшого набора системных вызовов, обеспечивающих непосредственный доступ к средствам ввода\вывода ОС. Одновременно с этим они являются основой для всей системы ввода\вывода *nix, и многие другие механизмы доступа к файлам (например) основаны именно на них.

Вот они: Open - открытие файла, Close - закрытие, Read - чтение данных, Write - запись данных, Lseek - перемещение в заданную позицию в файле, Fcntl - управление связанными с файлом атрибутами.

За что я люблю эту ОС, так это за то, что эти функции можно использовать почти для всего: и для работы с файлами, и для вывода на экран, и для организации обмена данными по сети. Один интерфейс для множества сходных задач. Сразу наплывают воспоминания о тех временах, когда я активно программировал под самой лучшей в мире операционной системой - Windows :).

Ближе к делу - рассмотрим элементарный рабочий пример:

```
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>

int main () {
  int fd;
  ssize_t nread;
  char buf[1024];
  fd = open("data", O_RDONLY);
  nread = read(fd, buf, 1024);
  close(fd);
};
```

Вызов Open открывает в текущем каталоге файл Data только для чтения, возвращая целочисленный (и неотрицательный) дескриптор файла, по которому система уже будет опознавать этот файл и предоставлять возможность выполнять с ним нужные действия. Далее идет системный вызов Read, читающий из файла с идентификатором fd первый килобайт данных. Все не так сложно.

Стоит заметить, что в случае возникновения ошибки любой из используемых выше системных вызовов вернет - 1. Чтобы узнать точное значение ошибки - подключить заголовочный файл errno.h и посмотреть значение переменной Errno. Или вызвать функцию Perror(), которая выведет текстовую интерпретацию ошибки на экран.

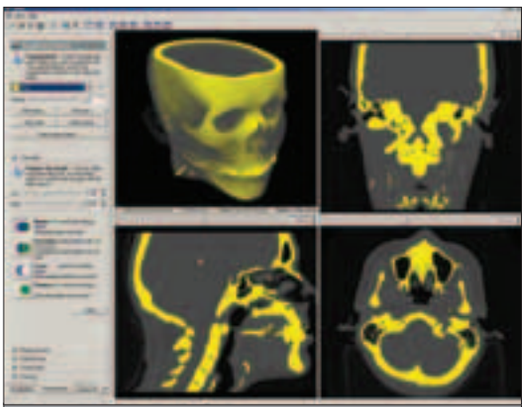
Внимание! Вызов Open имеет три параметра (последний - необязательный): строка, содержащая название файла, после которой идет целочисленный метод доступа. В этом случае использован O_RDONLY - только чтение. Также возможно использование O_WRONLY (только запись), O_RDWR (открытие для чтения и записи) или значение O_CREAT, используемое для создания файла. Как всегда, комбинировать значения можно при помощи "|" (например, O_CREAT | O_WRONLY). При использовании значения O_CREAT нужно передать системному вызову и третий параметр типа mode_t (на самом деле он тоже целочисленный), который будет характеризовать права доступа. Кстати, есть еще одно полезное значение второго параметра - O_TRUNC. При его использовании вместе с флагом O_CREAT файл будет усечен до нулевого размера (если он существует и если это позволяют права доступа).

Напоминаю, что после завершения работы файл нужно обязательно закрыть - системный вызов Close. Напоминаю и про существование man'ов по этим системным вызовам, в которых можно найти намного более подробную информацию.

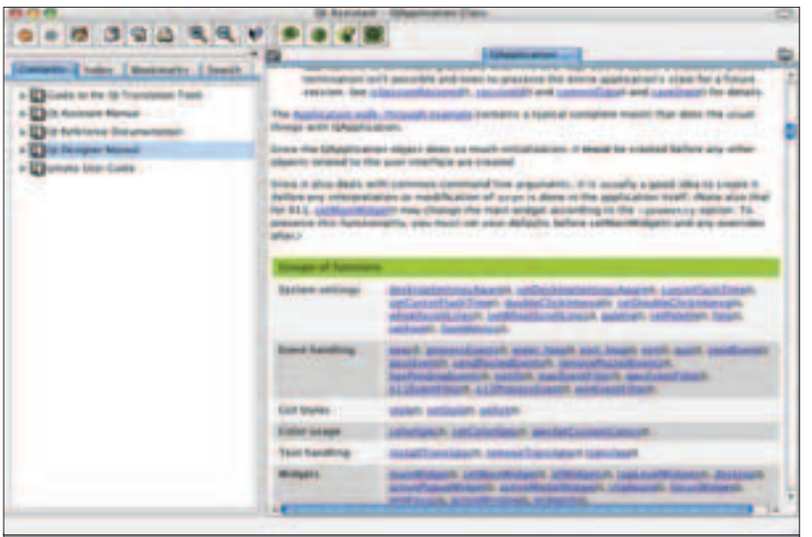
Не стоит забывать про функцию Fcntl, используемую для управления уже открытыми файлами. Она определена как int fcntl(int fd, int cmd, <что-то зависящее от cmd>). Больше других »

Читай, читай, читай и еще раз читай man'ы - если какая-то информация где-то от тебя прячется, то именно там.

Из сред разработки под *nix могу предложить таких монстров, как, например, KDevelop, Anjuta. Это из Иксовых, под КДЕ и GNOME соответственно.



Кроссплатформенные библиотеки в действии



Qt Assistent: с докумментацией у нас тоже все, в порядке :-)

интересны значения параметра cmd F_GETFL & F_SETFL. Они позволяют узнать\изменить текущие флаги статуса открытого файла. Вот как, например, можно узнать текущий статус файла (модификатор доступа):

```
int arg = fcntl(fd, F_GETFL);
if (arg & O_APPEND)
  printf("флаг O_APPEND");
if ((arg & O_ACCMODE) == O_RDWR)
  printf("файл открыт для чтения и записи")
```

Понятно, что текущий статус файла - некое число, отдельные биты которого сигнализируют об установке (или отсутствии) некоторых флагов. Как показано в примере, поле, в котором хранится значение модификатора доступа, можно вырезать с помощью специальной маски O_ACCMODE, определенной в fcntl.h.

При запуске новой программы ОС автоматически открывает три дескриптора файла, которые называются стандартным вводом, стандартным выводом и стандартным выводом диагностики соответственно. Они всегда имеют значения 0, 1 и 2. Не спутай stdin, stdout & stderr с чем-нибудь другим из стандартной библиотеки ввода\вывода.

По умолчанию использование системного вызова Read на стандартном вводе приведет к чтению с клавиатуры, запись в стандартный вывод или вывод диагностики приведут к выводу информации на экран терминала. Как ты понимаешь, такого может и не быть :).

Системные вызовы ввода\вывода являются основой всей системы ввода\вывода *nix, но они примитивны и предоставляют возможность работы с данными в виде простой последовательности байт.

Читай man'ы и посещай (очень советую) www.tldp.org, содержащий огромное количество самой разной увлекательной и познавательной информации по Linux.

Системные вызовы ввода\вывода являются основой всей системы ввода\вывода *nix, но они примитивны и предоставляют возможность работы с данными в виде простой последовательности байт, что не всегда может быть удобно для программиста, потому что заставляет его задумываться над многими вещами. Хочу напомнить тебе о стандартной библиотеке ввода\вывода, описанной в stdio.h и содержащей намного больше средств (fprintf, getc, putc, etc), нежели упомянутые системные вызовы.

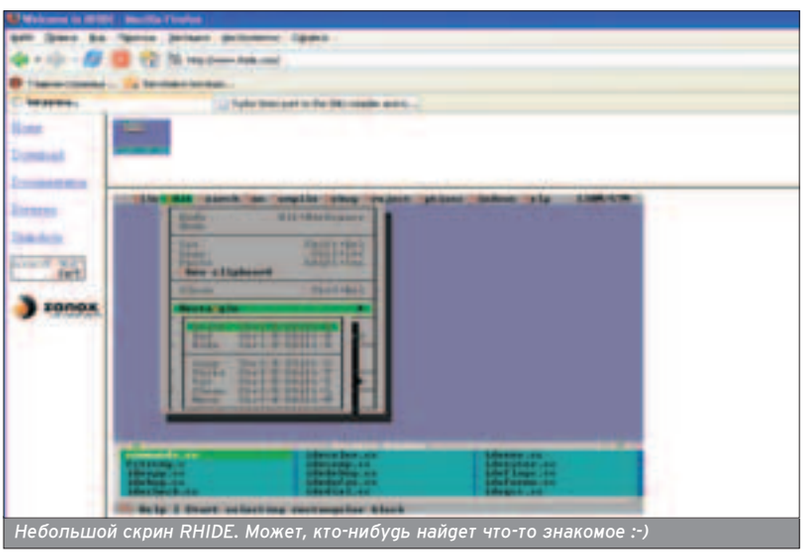
В *nix с каждым процессом связана маска создания файла, которая используется для автоматического выключения заданных битов прав доступа при создании новых файлов. Это бывает полезно для защиты от случайного включения "ненужных" прав доступа. В терминах языка С, если считать, что маска задана в целочисленной переменной Mask, то реальные права доступа будут получены следующим выражением: (~mask) & mode.

Для изменения маски создания файла существует системный вызов Umask, принимающий единственный параметр типа mode_t (который в очередной раз оказывается обычным целым числом). Например, вызов umask(022) запрещает вновь созданным файлам текущего процесса присвоение файлу прав доступа на запись. Всем, кроме владельца.

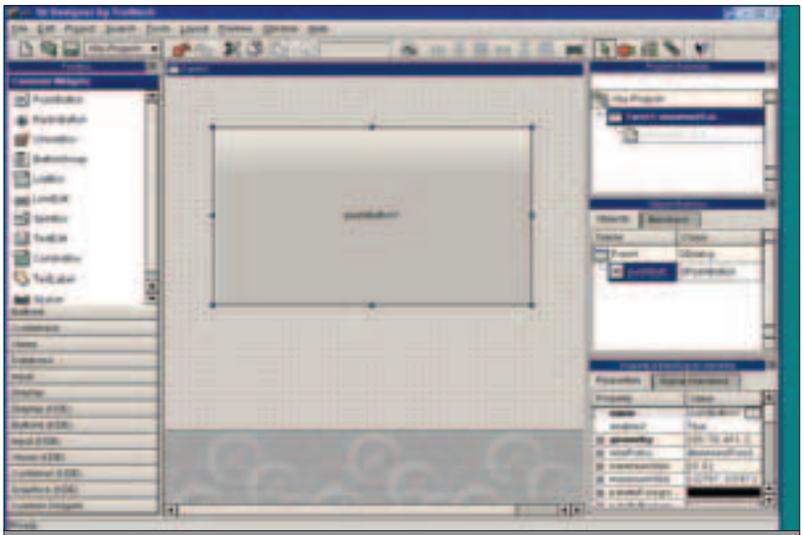
Раз речь зашла о правах доступа, и теперь стоит упомянуть такой системный вызов, как Access, который определяет, может ли процесс получить доступ к файлу в соответствии с истинным (а не с действующим) идентификатором пользователя и группы. И еще один системный вызов - Chmod. Здесь комментарии излишни.

В *nix один файл может иметь несколько имен. То есть существует возможность связать одну и ту же последовательность данных с несколькими именами, а создавать копии файла не нужно. Такое имя называется жесткой ссылкой, а количество таких ссылок, связанных с файлом, - счетчик ссылок. Для добавления нового имени используется системный вызов Link (const char *original, const char *link), а для удаления - Unlink (char *name), который просто удаляет указанное имя и уменьшает счетчик ссылок на единицу. Сами же данные будут безвозвратно потеряны только в том случае, если этот счетчик равен

KDevelop



Небольшой скрин RHIDE. Может, кто-нибудь найдет что-то знакомое :-)



Создаем UI с помощью Qt Designer

нулю и если указанный файл не открыт для чтения ни в одной программе. Замечу, что для создания символьных ссылок используется системный вызов Symlink, описанный в unistd.h.

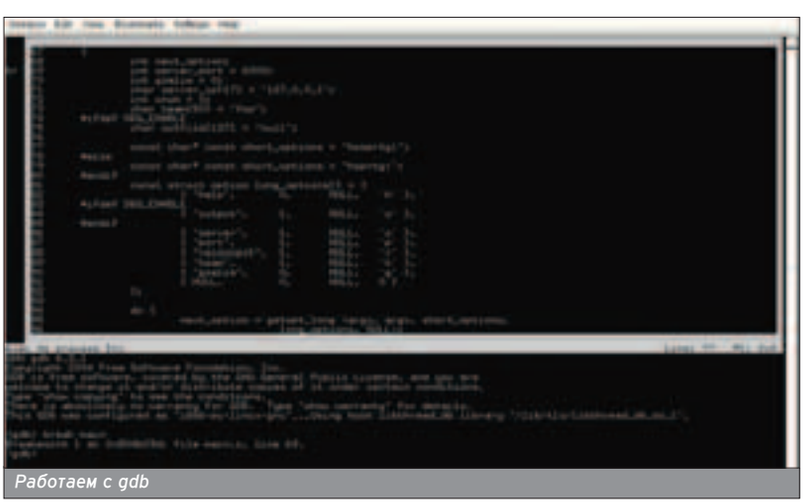
МНОГОЗАДАЧНОСТЬ

Время однозадачных ОС уже прошло, и большинство современных операционных систем - многозадачные. К ним можно применять понятие процессов, некоторое количество которых может одновременно выполняться в текущий момент времени.

Основным примитивом, создающим процесс в *nix-системах, является системный вызов Fork() - фактически в нем и заключается вся многозадачность ОС. После успешного вызова этой функции ядро создаст новый, почти идентичный текущему процесс - дочерний. Оба процесса будут выполняться одновременно, продолжая свое выполнение с оператора, следующего за Fork(). Как различить эти два процесса? Если функция возвращает 0, значит, ты добрался до процесса-потомка. Если возвращаемое число > 0, то ты застрял на родительском. Иначе - системный вызов завершился с ошибкой, которую нужно анализировать.

Следующим системным вызовом, который необходимо упомянуть - Execve, а точнее целое семейство вызовов, которые в итоге сводятся к одному, упомянутому мной. Выполняемая функция данного множества системных вызовов одна - загрузка новой программы в пространство памяти процесса. После одного из таких вызовов ни одного оператора, следующего за ним, выполнено не будет. Конечно, если и сюда ошибка не добралась своими грязными руками.

Кроме процессов существует еще один вид программной сущности - потоки. Что-то вроде облегченных процессов, выполняющихся в одном адресном пространстве. Примитивов работы с ними немало, и для каждой системы они свои. Однако любая POSIX-система включает в себя реализацию Posix-threads. Пару слов о ней: в ОС есть набор функций, описанных в pthread.h и называющихся pthread_*. Например, pthread_create() создает новый поток, возвращая его идентификатор. Конечно, одним из параметров этой функции будет указатель на функцию, которая станет новым потоком и начнет свое выполнение. Функция Pthread_cancel может попытаться завершить поток, а Pthread_exit будет являться аналогом обычного Exit для процесса.



Работаем с gdb

NB: после создания нового потока он сам (новый поток) и процесс, породивший его, начнут выполняться параллельно. И кто их знает, кто завершится первым. Для этого и существует функция Pthread_join, которая ждет завершения определенного потока и заодно запоминает значение, которое он вернул по завершению. Все примеры использования этих функций ты найдешь в соответствующих man'ах. Главное не испугаться и заглянуть в них: все очень просто! Стоит только внимательно прочитать и один раз правильно применить полученную информацию.

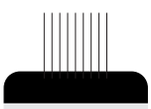
TURN OFF

К моего большому сожалению, не могу подробно описывать все аспекты программирования на C под *nix, но я постарался упомянуть ключевые слова и направления, по которым будет устроен дальнейший поиск информации. Кроме man'ов, я очень советую тебе посетить www.tldp.org, в котором огромное количество самой разной интересной информации по Linux. Первым делом советую заглянуть в NCurses programming guide, описывающую работу с библиотекой Ncurses, позволяющей гораздо более удобно взаимодействовать с терминалом и создавать текстовый пользовательский интерфейс. Далее я бы порекомендовал почитать про *unix IPC, межпроцессорное взаимодействие - очереди сообщений, семафоры и пайпы.

Если же не хочешь заморачиваться с ворохом всех этих системных вызовов и библиотек, всяких там файлов и терминалов, а желание быстро, легко и не отвлекаясь на уточнение тонкостей писать программы есть - читай про такие библиотеки, как Qt, GTK и т.д., которые вдобавок позволят использовать графический интерфейс. Окошки там всякие, кнопочки.

В общем, надеюсь, своим повествованием не отбил у тебя интерес к этой теме.

Linux Programming Guide (www.tldp.org/LDP/lpg)



Основным примитивом, создающим процесс в *nix-системах, является системный вызов Fork() - фактически в нем и заключается вся многозадачность ОС.

Андрей Семенюченко (semu@rbcmail.ru)

ШЕЛЛ ДЛЯ КОДЕРА

ПРОГРАММИРУЕМ НА BASH. РАЗБОР РЕАЛЬНОГО СЦЕНАРИЯ

Можно сколько угодно спорить о том, какой язык программирования лучше, но нельзя спорить только с тем, что необходимо использовать то, что позволит тебе добиться достойного результата при наименьших затратах времени и сил. Программирование на скриптовых языках, с одной стороны, является достаточно простым и понятным, а с другой - достаточно гибким и мощным средством для решения многих повседневных задач.

Языки сценариев полезны для пользователя и просто жизненно необходимы для любого системного администратора. *nix-системы имеют множество встроенных и прикладных языков программирования. Наиболее популярными и часто используемыми из них являются Perl, Tcl, а также shell. Сейчас ты, наверное, подумал: "Ха, да ведь шелл - это же командная оболочка, являющаяся как бы посредником между человеком и системой для упрощения взаимодействия". Совершенно верно, но не только! Это еще и мощное средство программирования. Плюсы использования интерпретируемых языков программирования очевидны. Вот только некоторые из них.

❶. Переносимость: ты можешь легко залить свой только что испеченный скрипт с машины, на которой установлена твоя любимая Fedore Core, на любую другую платформу под управлением, скажем, FreeBsd или Solaris. Главное, чтобы в системе был установлен интерпретатор для языка, на котором написан крипт.

❷. Простота написания кода: нет необходимости специально обучаться сложному программированию на компилирующих языках, таких как C, C++, Pascal, Fortran. Такое программирование наиболее очевидно, поскольку программа пишется в пошаговом режиме, то есть человек принимает решение о своем следующем шаге в зависимости от реакции системы на предыдущий шаг.

❸. Быстрота написания кода: благодаря простоте синтаксиса и отладки ты сэкономишь много времени.

❹. Большие функциональные возможности: хотя интерпретируемые языки и нельзя сравнить по своей функциональности, например, с C, тем не менее, не нужно недооценивать всей их мощи.

ПРОГРАММИРОВАНИЕ НА SHELL

Давай рассмотрим программирование на shell более подробно. Поскольку у тебя на Linux определенно есть sh и, скорее всего, bash, то нет никакой необходимости устанавливать пакеты этих программ, а можно сразу же приступить к программированию. Сразу же оговорюсь: в своих экспериментах я использовал интерпретатор bash.

Нет ничего страшного, если в качестве оболочки ты используешь sh. Возможно также, что у тебя установлен Korn Shell (ksh) или что-то еще, тогда тебе нужно всего лишь придерживаться стандарта POSIX, если хочешь, чтобы твои shell-сценарии могли быть интерпретированы другим шеллом. Пройдя по ссылке www.unix.org.ua/orelly/unix/ksh/appa_02.htm, ты сможешь прочитать статью о IEEE 1003.2 POSIX shell стандарте и его поддержке в Korn shell. Дополнительные же преимущества bash опишу чуть позже.

Важно понять, что из сценариев доступны абсолютно все команды и утилиты системы, а внутренние команды shell - условные операторы, операторы циклов и др. только увеличивают мощь и гибкость сценариев.

Обычно все сценарии начинаются с одной из следующих строк или набора строк:

```
#!/bin/sh
#!/bin/bash
#!/usr/bin/perl
#!/usr/bin/tcl
#!/bin/sed -f
#!/usr/awk -f
```

Ты, наверное, уже заметил, что каждая строка начинается одинаково, с символов "#!". Эти строки объясняют системе, что запущенный файл, - это не что иное, как сценарий, и его следует обработать с помощью указанного после символов "#!" интерпретатора.

Запустить сценарий можно двумя способами. Первый заключается в предоставлении права на исполнение файла для владельца файла, группы или всех пользователей в системе. Полный доступ к файлу на выполнение, запись и чтение для владельца выглядит так:

```
chmod 700 my_script_name.sh
```

или

```
chmod u+rwx my_script_name.sh
```

IEEE 1003.2 POSIX shell стандарт

Список сценариев в /etc/rc.d/init.d

Доступ для всех остальных, я думаю, ты сможешь сделать сам. Второй способ заключается в указании интерпретатора перед именем сценария:

```
sh my_script_name
bash my_script_name
```

ОТ ТЕОРИИ К ПРАКТИКЕ

Я думаю, наступил момент рассмотреть какой-нибудь пример практического использования скрипта, из которого сразу все станет ясно. Но перед этим хочу обратить твое внимание на то, что некоторую полезную роль в написании кода играют редакторы. Современные редакторы имеют уйму разных функций от подсветки кода до вставки в текст некоторых несложных конструкций. Все это на любителя. Ты можешь использовать vi или emacs, а мне хватает встроенного в Midnight comander редактора с подсветкой синтаксиса. Ну а выбор, как всегда, за тобой. Определись, что тебе больше нравится, и забудь.

Чтобы далеко за примерами не ходить, я решил залогиниться на свой ALT Master 2.2 и взять первый попавшийся на глаза скрипт запуска одного из демонов. Такие скрипты, как известно, находятся в /etc/rc.d/init.d/. Первым в каталоге init.d оказался файл Anacron - сценарий запуска для одного очень популярного планировщика задач, похожего на всем известный демон cron. Anacron также может периодически запускать команды в назначенное время, и в отличие от cron, нет необходимости постоянной работы системы (но это уже совсем другая история).

Разберем файл построчно. Как мы видим, в начале файла используется до боли знакомая конструкция, начинающаяся с "#!". Ты уже знаешь, на что она указывает системе. Далее следуют комментарии, которые предваряются символом "#". А вот тут уже становится интересно: появилась какая-то строка, да еще с точкой в начале. Вот она:

```
. /etc/init.d/functions
```

На самом деле ничего странного здесь нет. Символ "." является эквивалентом команды source. Внутри сценария команда source other_file_name подключает файл other_file_name. Она очень напоминает директиву препроцессора языка C/C++ - "#include". Коротко пробегись по включенному файлу и получи представление о том, что же представляет собой скрипт functions. На самом деле все становится предельно ясно с комментариями к файлу. Этот сценарий содержит функции, наиболее часто используемые скриптами автозапуска из /etc/init.d. Дальше в файле как раз встречаются функции, которые очень часто можно найти в скриптах.

Но вернемся к нашему сценарию автозапуска anacron.

```
[ -f /usr/sbin/anacron ] || exit
```

В данной строке мы видим опять же эквивалент команды Test. Как понятно из названия, она проверяет условие, которое в этом случае заключено в квадратные скобки. Ключ -f задается для проверки существования файла. Таким образом, данный блок операторов служит для того, чтобы точно знать, существует ли файл демона /usr/sbin/anacron, и только в этом случае продолжать выполнение скрипта, а иначе выйти вон.

Далее следует инициализация переменных LOCKFILE и RETVAL, которая происходит при присвоении им определенных значений. Пока для нас эти переменные ничего не значат. При программировании на shell переменные не имеют типа, но в зависимости от того, какое значение им присвоено, возможна, например, целочисленная арифметика с переменными. После того как переменной присвоено значение, ее можно использовать в качестве подстановки, приписав в начале ее имени символ "$". И помни разницу между именем переменной (RETVAL) и ее значением ($RETVAL): если, например, посмотреть в самый конец рассматриваемого скрипта, обнаружишь строку exit $RETVAL. Здесь используется оператор exit для завершения программы, который тоже возвращает значение переменной RETVAL.

Ну вот мы и добрались до начинки файла - объявления функций start(), stop() и restart(). Под их контроль как раз и попадает обработка параметров, поступающих скрипту от пользователя или других программ. Как понятно из названий, каждая функция производит соответственно запуск, останов или restart демона. В принципе, здесь все понятно. Интересно то, что дальше в функциях встречаются не совсем логичные переменные $?, $$, $PPID. Ничего подобного не объявлялось, тогда откуда они взялись? Сейчас все станет ясно. Дело в том, что существует специальный тип переменных - так называемые переменные окружения. В рамках любого процесса есть некоторое окружение, то есть набор переменных, к которым он может обращаться за получением определенных данных. Каждый раз, когда запускается командный интерпретатор, для него создаются переменные окружения. Эти переменные можно экспортировать любому дочернему процессу с помощью команды Export. Список переменных можно получить командой Set. Количество переменных окружения достаточно велико, поэтому в командной строке лучше дать команду set|more для того, чтобы иметь возможность пролистать весь список.

Так вот, переменная $? содержит значение последней выполнившейся команды. А переменная $$ таит в себе не что иное, как PID сценария, то есть идентификатор процесса сценария. Переменная $PPID - PPID, то есть родительский идентификатор процесса.

Получается вот что (сразу не скажу, что): внимательно посмотри на функцию Start().

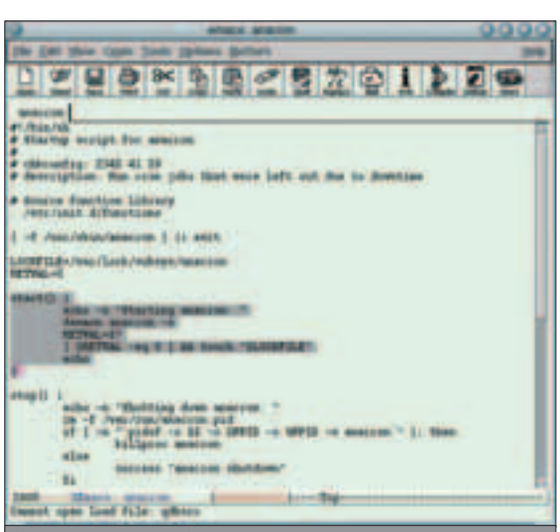
```
daemon anacron -s
```

Командой Daemon пытаемся запустить файл демона anacron с опцией -s для синхронизации заданий. При удачном запуске команда Daemon вернет значение "0".

Уже известно, что переменная $? будет содержать код возврата пос- »

Скачать tar.gz архив с исходниками bash любой версии, а также прочитать более подробную информацию ты можешь с сайта GNU Project www.gnu.org/software/bash/bash.html

Нужно учитывать, что строка #!/bin/sh на самом деле означает интерпретатор, использующийся в системе по умолчанию, которым в большинстве дистрибутивов Linux является bash.



А так выглядит файл при использовании xemacs

ледней операции. Это значение и присваиваешь переменной RETVAL.

```
[ $RETVAL -eq 0 ] && touch "$LOCKFILE"
```

И теперь просто проверяешь, запущен ли демон, и только в этом случае создаешь файл, содержащийся в переменной LOCKFILE, который необходим для работы демона.

С функцией stop() все абсолютно то же самое. Проверяешь, действительно ли запущен демон, и если все благополучно, вызываешь команду Killproc, которая убивает демона. И подчищаешь созданные ранее файлы.

Функция restart() поочередно вызывает функции start() и stop(), чтобы перезапустить демон.

Ниже в файле расположен оператор выбора case, который выполняет тот или иной участок кода согласно заданным условиям. Иногда case называют блоком операторов, поскольку его можно представить в качестве большого количества операторов проверки условия if, then, else. В нашем случае проверяется значение переменной $1, которое представляет собой не что иное, как первый параметр, передающийся скрипту. Например, если в командной строке набрать anacron start, выполнится условие start, которое вызовет функцию start() и запустит демон. Блок case завершает ключевое слово esac.

При работе со строками в bash можно воспользоваться неинтерактивным строчным редактором sed и языком обработки шаблонов awk.

Определить версию bash, установленную у тебя, можно с помощью параметра --version.

ОТЛАДКА SHELL-СЦЕНАРИЕВ

Нужно признаться, что до выхода последней версии (bash 3.0) командный интерпретатор bash не имел своего отладчика и даже каких-либо отладочных команд, возможно, за исключением команды Trap, которая устанавливает ловушки на сигналы, то есть определяет, какие действия нужно выполнить при получении сигнала. Формат команды Trap следующий:

```
trap [-lp] [arg] [sigspec ...]
```

Команда Arg выполняется при получении командным интерпретатором указанных сигналов sigspec. Если указана опция -p, выдаются команды Trap, связанные с каждым из перечисленных сигналов. Опция -l приводит к выдаче списка имен сигналов и соответствующих им номеров. Сигнал можно задавать как по имени, определенному в файле <signal.h>, так и по номеру. Если в качестве сигнала указано DEBUG, команда Arg выполняется после каждой простой команды. Trap возвращает 0 в случае успеха, в противном случае -1.

Могут пригодиться некоторые команды, которые, казалось бы, к отладке не имеют никакого отношения. Например - команда Echo, которая умеет выводить значения переменных в процессе выполнения скрипта.

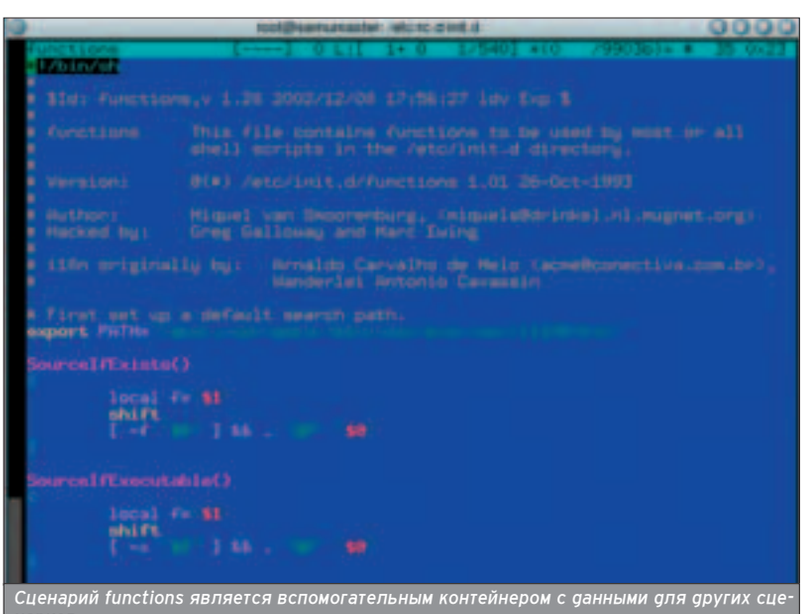
А если хочешь чего-то более продвинутого, то в целях отладки можешь воспользоваться командой Tee и функцией assert(). Tee проверяет процессы и потоки данных в опасных ситуациях, а функция assert() служит для проверки переменных и условий в указанных точках сценария.

Конечно же, ты помнишь, что скрипт можно запустить строкой вида:

```
bash my_script_name
```

Если интерпретатору передать аргументы -n, -v или -x перед именем сценария, можно еще и получить некоторую полезную информацию. Ключ -n проверяет наличие синтаксических ошибок не запуская сам скрипт, ключ -v выводит каждую команду до того как она будет выполнена, -x показывает результаты выполнения команд.

Например, если добавить любую некорректно сформированную строку в данный скрипт так, чтобы он не мог запуститься из-за синтаксической ошибки. Я просто добавил выражение "This is error for test" сразу после



Сценарий functions является вспомогательным контейнером с данными для других сценариев

КОД СЦЕНАРИЯ ANACRON

```
#!/bin/sh
# Startup script for anacron
#
# chkconfig: 2345 41 59
# description: Run cron jobs that were left out
due to downtime

# Source function library.
. /etc/init.d/functions

[ -f /usr/sbin/anacron ] || exit

LOCKFILE=/var/lock/subsys/anacron
RETVAL=0

start() {
    echo -n "Starting anacron: "
    daemon anacron -s
    RETVAL=$?
    [ $RETVAL -eq 0 ] && touch "$LOCKFILE"
    echo
}

stop() {
    echo -n "Shutting down anacron: "
    rm -f /var/run/anacron.pid
    if [ -n "`pidof -o $$ -o $PPID -o %PPID -x
anacron`" ]; then
        killproc anacron
    else
        success "anacron shutdown"
    fi
    RETVAL=$?
    [ $RETVAL -eq 0 ] && rm -f "$LOCKFILE"
    echo
}

restart()
{
    stop
    start
}

# See how we were called.
case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    reload|restart)
        restart
        ;;
    condstop)
        if [ -e "$LOCKFILE" ]; then
            stop
        fi
        ;;
    condrestart)
        if [ -e "$LOCKFILE" ]; then
            restart
        fi
        ;;
    status)
        status anacron
        RETVAL=$?
        ;;
    *)
        echo "Usage: ${0##*/} {start|stop|reload|
restart|condstop|condrestart|status}"
        RETVAL=1
esac

exit $RETVA
```

BASH 3.0 - ДВА ГОДА СПУСТЯ

Почти два года прошло после выхода предыдущей версии bash (2.0). Видимо, разработчики зря времени не теряли: в недавно вышедшей версии bash 3.0 появилось много чего нового. Исправлены многочисленные ошибки, улучшена поддержка многобайтовых символов, устранены некоторые несовместимости с POSIX, интегрирована возможность интернационализации сообщений и отладчик bash debugger.

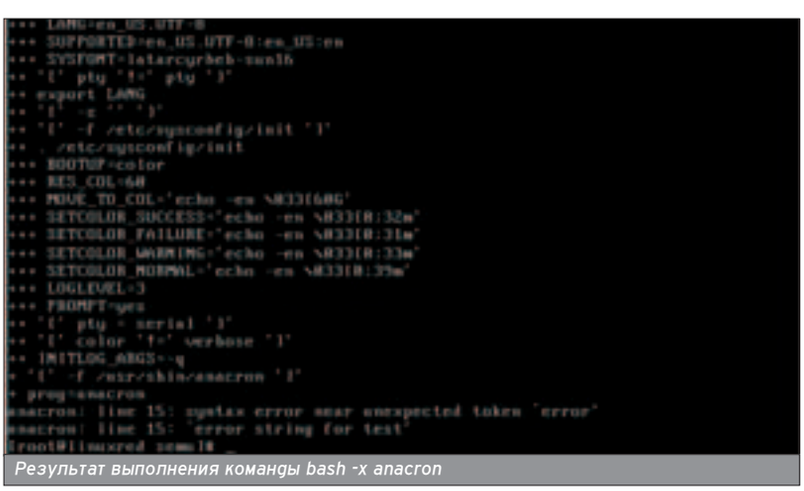
Вот только неполный список нововведений в bash 3.0:
- расширение ANSI. Появилась возможность задания escape-последовательности в шестнадцатеричном виде \x{hexdigits};
- новая переменная COMP_WORDBREAKS, которая содержит набор символов для разделения слов;
- изменение формата записи символов в значении переменной HISTCONTROL;
- поддержка многобайтовых символов с появлением нового аргумента --enable-multibyte, который нужно указать для configure при установки bash;
- новые переменные для реализации встроенного отладчика bash: BASH_ARGC, BASH_ARGV, BASH_SOURCE, BASH_LINENO, ASH_SUBSHELL, BASH_EXECUTION_STRING, BASH_COMMAND;
- для операторов for, case, select, arithmetic commands теперь хранится специальная информация, используемая для отладки;
- новая ловушка RETURN, используемая при возвращении функции;
- новая опция bash для так называемой внешней отладки - debugger;
- добавлены операторы для упрощения и оптимизации применения регулярных выражений;
- интегрированы пакет Gettext и библиотека Libintl для перевода сообщений шелла на различные языки.

Полный перечень исправлений и новых функций ищи здесь: **http://cnswww.cns.cwru.edu/~chet/bash/NEWS**

объявления функции start(). Теперь запустим его следующим образом:

```
bash -x anacron
```

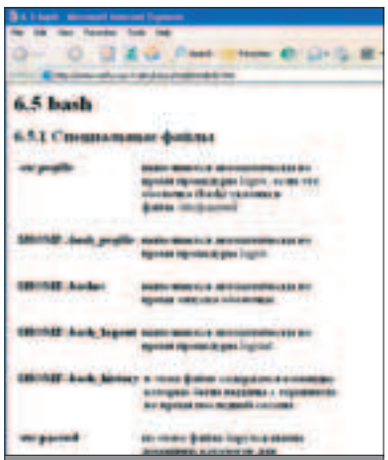
В результате получили результаты выполнения команд, а также сообщение о синтаксической ошибке в строке №15 сценария.



Результат выполнения команды bash -x anacron

ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ BASH

На самом деле возможности Bash сильно отличаются от умений многих других интерпретаторов, именно поэтому этот интерпретатор в настоящее время стал стандартом де-факто в Linux-системах. Другим не менее распространенным шеллом является sh, который по умолчанию поставляется



Файлы, используемые для настройки и работы bash

со всеми системными дистрибутивами. Но по удобству использования и по функциональности sh сильно уступает bash, поэтому последний набирает все больше голосов как среди программистов и администраторов, так и среди рядовых пользователей. Ниже приведен список (далеко не полный!) возможностей, имеющихся у bash 2.0 и отсутствующих у sh:
- наличие оператора выбора select;
- зарезервированное слово function для оформления функций;
- ведение истории команд;
- специальный синтаксис оператора цикла for для использования совместно с арифметическими операциями: for ((expr1 ; expr2; expr3)); do list; done;
- перенаправление вывода: <>, &>, >|;
- режим posix для изменения поведения команд согласно стандарту;
- расширение регулярных выражений для выполнения действий с подстроками (${p%[%]w}, ${p#[#]w});
- ловушка DEBUG trap;
- ловушка ERR trap;
- egrep-подобное расширение для поиска значения по образцу;
- возможность поиска без учета регистра;
- перенаправление в /dev/fd/N, /dev/stdin, /dev/stdout, /dev/stderr, /dev/tcp/host/port, /dev/udp/host/port

Кроме того, bash имеет большое количество полезных переменных окружения: BASH, BASH_VERSION, BASH_VERSINFO, UID, EUID и т.п.

В редакторе xemacs (emacs под X-Windows), помимо удобных средств редактирования, присутствуют также дополнительные средства отладки сценариев.

Опция -posix изменяет поведение bash в соответствии со стандартом 1003.2.

ПОДВОДИМ ИТОГИ

Как всегда после любой работы, нужно собрать урожай. А его немало: подробно рассмотрели код сценария автозапуска демона-планировщика anacron, убедились в силе и простоте программирования на скриптовых языках, узнали о нововведениях в bash 3.0 и об отладке только что написанного скрипта даже при отсутствии специальных возможностей отладки в старых версиях bash. Надеюсь, теперь ты избавился от фобии правки кодов имеющихся скриптов и даже сам можешь сварганить нечто подобное без особых проблем. Удачи!

Крис Касперски aka мыщъх

ИЗ WINDOWS В *NIX

ПОСОБИЕ ПО ПОРТИРОВАНИЮ ПРИЛОЖЕНИЙ

Последнее время много разговоров о переносе *nix-программ на Windows. Только так, но никак не иначе. А ведь существует большое количество Windows-программ, аналога которым на других платформах не существует (прежде всего это твои собственные программы). Стоит ли их переносить на *nix, и если да, то как?

...задачи, решаемые с помощью компьютера, нередко самим компьютером и порождаются.
Пол Грэм

Абсолютного переносимого программного обеспечения не существует, как не существует абсолютного нуля. Понятие "переносимость" еще не означает, что портирование сводится к простой перекомпиляции. Всегда требуются дополнительные усилия по его адаптации. Иногда эти усилия настолько значительны, что проще переписать программу с нуля, чем гонять ее между платформами. Системно-ориентированные пакеты (FAR, soft-ice) переносить вообще бессмысленно.

В любом случае ты должен полностью разобраться в исходных текстах, которые переносишь. При доминирующем стиле кодирования интерфейс программы перемешан с "вычислительной" частью (спасибо визуальными средам разработки!), и разделить их не проще, чем сиамских близнецов (но разделять все же придется). Типичный код нашпигован большим количеством системно-зависимых функций: вместо стандартных библиотечных функций преобладают вызовы API и MFC. Активно используются ассемблерные вставки и повсеместно - умолчания компилятора. Это в Borland'e char по умолчанию unsigned, а в других компиляторах он ведет себя совсем не так! Про "умолчанную" кратность выравнивания структур я вообще молчу. Хуже только нестандартные расширения компилятора и специфические особенности его поведения. Большинство программ, созданных современными "программистами", не компилируются MS VC, если написаны на BCC и, соответственно, наоборот. До переноса на *nix им так же далеко, как их авторам до звания "программиста" (необязательно даже "почетного программиста", можно просто "стажера", путающего язык со средой разработки).

Считается, что перенос сокращает издержки на развитие и сопровождение проекта. Имея независимые версии для Windows и *nix, ты вынужден вносить исправления и гонять баги в обеих программах одновременно. Портабельный код этих недостатков лишен. Якобы. Скажи, когда-нибудь ты пробовал писать программу, компилируемую более чем одним компилятором? Ругался при этом? И правильно! Я бы тоже ругался. Ограничения, налагаемые переносимым кодом, лишают нас многих возможностей языка и значительно увеличивают трудоемкость разработки. Допустим, ты используешь шаблоны (Templates), и на MS VC все работает, но при переходе на другой компилятор программа разваливается к черту. А некоторые компиляторы не инициализируют статические экземпляры класса. Ну не инициализируют и все тут! Забудь о стандартах. Компиляторы все равно их не придерживаются.

При каждом внесении изменений в программу прогоняй ее через все целевые компиляторы. Код, специфичный для данной платформы, заботливо окружи #ifdef или вынеси в отельный файл, ну и т.д. В конечном счете ты получишь все те же два независимых проекта, но тесно переплетенные друг с другом, причем внесение изменений в один из них дает непредсказуемый эффект в другом. Нет-нет, не подумай! Я вовсе не противник переносимого кода, просто не понимаю тех, для кого переносимость является целью, а не средством. Никто не спорит, что такие проекты, как Apache или GCC, должны изначально разрабатываться как переносимые (процент системно-независимого кода в них очень велик), но вот мелкую утварь типа почтового клиента лучше затачивать под индивидуальную платформу, а при переходе на *nix переписывать заново.

СЛОИ АБСТРАГИРОВАНИЯ

Если нужно быстро перенести программу - воспользуйся WINE или Willows. Это бесплатно распространяемые имитаторы Windows, оборачивающие *nix-функции толстым слоем переходного кода, реализующего Win32 API и работающие на: Windows 9x/NT/2000/XP, Linux, FreeBSD, Solaris, а Willows еще и на QNX.

Обрати внимание: не эмуляторы, а именно имитаторы (WINE именно так и расшифровывается: Wine Is Not Emulator - это вам не эмулятор). Портируемая программа исполняется на "живом" процессоре практически без потерь в скорости. Во всяком случае реклама говорит именно так. А что реальная жизнь? При всей схожести *nix и Windows NT (их ядра наследуют общий набор концепций) они во многом различаются. В *nix есть замечательная функция Fork, расщепляющая процесс напополам. В NT ее нет. Функциям CreateProcess/CreateThread далеко до Fork. И вот почему. Накладные расходы на расщепление процесса ничтожны, чего нельзя сказать о создании процесса/потока с нуля. Кстати, с потоками в Linux сплошной напряг; внутренние потоки представляют те же процессы, но только немного усложненные. Всегда заменяй CreateThread на Fork, когда это только возможно (процессы, в отличие от потоков, исполняются в различных адресных пространствах и могут обмениваться данными только через IPC, например, так происходит с проецированными в память файлами). К тому же средства синхронизации потоков в Windows и *nix далеко не как две капли воды, а в Linux-син-

При каждом внесении изменений в программу прогоняй ее через все целевые компиляторы.

Windows-приложение, запущенное под WINE

хронизация не поддерживается вовсе и реализуется внешними библиотеками. Все это делает отображение Win32 API на *nix-функции неоднозначным, и выбор предпочтительного системного вызова в каждом конкретном случае должен определяется индивидуально. Человеком. Имитатор на это не способен, и падения производительности не избежать (другое дело, что при современных аппаратных мощностях о производительности можно не вспоминать).

Конструктивно большинство имитаторов состоят из двух основных компонентов: бинарного интерфейса (Binary Interface) и библиотеки разработчика (Library). Некоторые имитаторы (например, Willows) включают еще и уровень абстрагирования от платформы (Platform-abstraction Layer), что упрощает их перенос на другие системы, но это уже детали реализации.

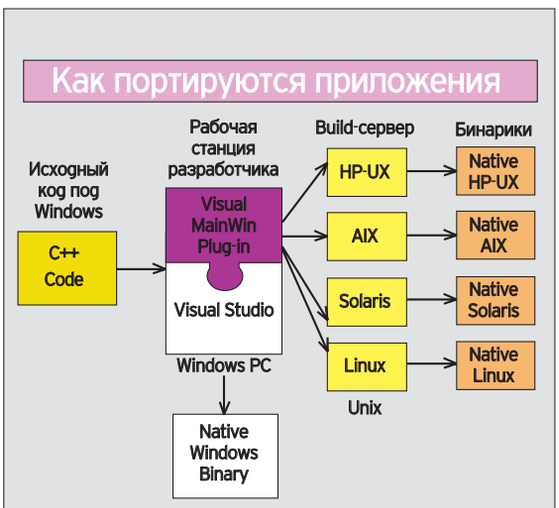
Бинарный интерфейс включает в себя win32-загрузчик, "перевариваю-щий" PE-файлы и с максимальной точностью воссоздающий привычное для них окружение. Необходимость в перекомпиляции при этом отпадает, однако совместимость остается на очень низком уровне. Реально удается запустить лишь небольшое количество офисных приложений типа Office, Acrobat или Photoshop. Системные утилиты, скорее всего, откажутся работать, и тут на помощь приходит библиотека - заголовочные файлы плюс lib-файл. Адаптировав приложение, можно компилировать его как в ELF (тогда необходимость иметь на машине установленный имитатор отпадает) или в PE. Красота!

В крайнем случае можно воспользоваться полноценным эмулятором PC - VMWare или Win4Lin, однако полезность этого решения сомнительна. Дело даже не в аппаратных требованиях (я вполне успешно гоняю VMWare на P-III 733), а в удобстве использования (точнее, его отсутствии). Достаточно сказать, что обмениваться данными с эмулятором придется через виртуальную локальную сеть, гоняя их в обе стороны, в хвост и в гриву.

ПЕРЕНОС ПРИЛОЖЕНИЙ, СОЗДАННЫХ В MICROSOFT VISUAL STUDIO

Компания Mainsoft (та самая, у которой свистнули исходные тексты Windows 2000) выпустила замечательный продукт Visual MainWin, позволяющий писать код в Microsoft Visual Studio и тут же компилировать его под разные платформы (Windows, Linux, HP-UP, AXI, Solaris), причем количество поддерживаемых платформ постоянно растет.

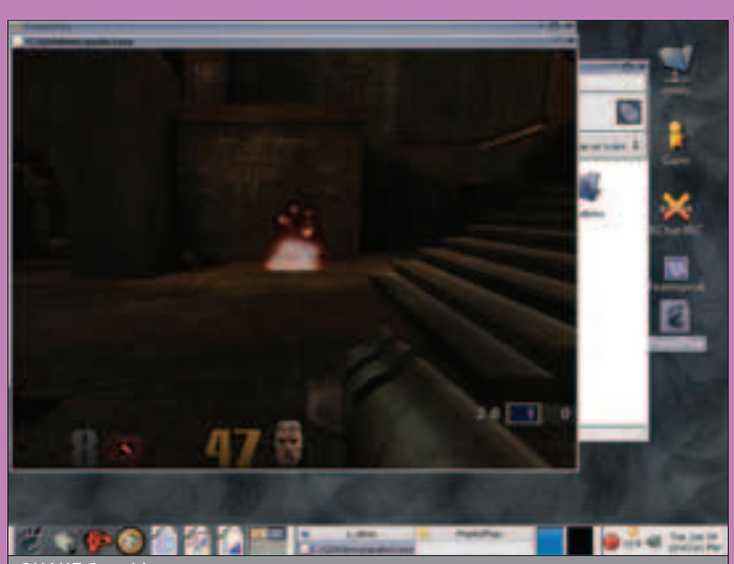
Пакет состоит из нескольких частей - это и инспектор кода, позволяющий обнаружить системно-зависимые



Портирование приложений под Visual WinMain, интегрированного в Microsoft Visual Studio

УСТАВШИМ ОТ "ПАСЬЯНСА" ПОСВЯЩАЕТСЯ

Для переноса игр и других графических приложений лучше всего подходит WineX, в настоящее время переименованный в Cedega, - коммерческая версия имитатора WINE от компании Transgaming, ориентированная на DirectX, Direct3D, OpenGL и прочие технологии этого уровня. Работает в Linux, Mac, PlayStation 2, XBox и Next Gen. Хочешь "поквакать" в Linux? Нет проблем! А еще можно "подумать" или погонять в Need-for-Speed. Список поддерживаемых игр очень и очень велик.



QUAKE 3 на Linux

участки (пускай программист сам решает, как он будет их исправлять!), и препроцессор, подготавливающий исходный код к последующей трансляции GCC (или любым другим *nix-компилятором), и, конечно же, обширная библиотека функций, реализующая: а) Windows-примитивы (SEH, DLL, процессы/потоки, средства их синхронизации, реестр, буфер обмена и поддержку национальных языков); б) графический и пользовательский интерфейс (GDI32, USER32); в) COM-модель (ActiveX, OLE, MIDL, DCOM); г) библиотеку времени исполнения (ALT, MFC, C Runtime library). Полный перечень на www.mainsoft.com/solutions/vmw5_wp.html.

Это коммерческий продукт, причем очень коммерческий (лицензия на одного разработчика стоит больше $2000), правда, доступна 30-дневная полнофункциональная демо-версия. Так что решай сам: нужно оно тебе или нет.

MainWin, конечно, мощная штука, но иногда требуется приложение помельче. Основной камень преткновения - это, конечно же, MFC. В Microsoft Visual Studio все визуальные средства разработки построены именно на нем. И хотя исходные тексты MFC доступны, перенести его на *nix-системы намного сложнее, чем создать с нуля, сохранив иерархию классов и прототипы функций.

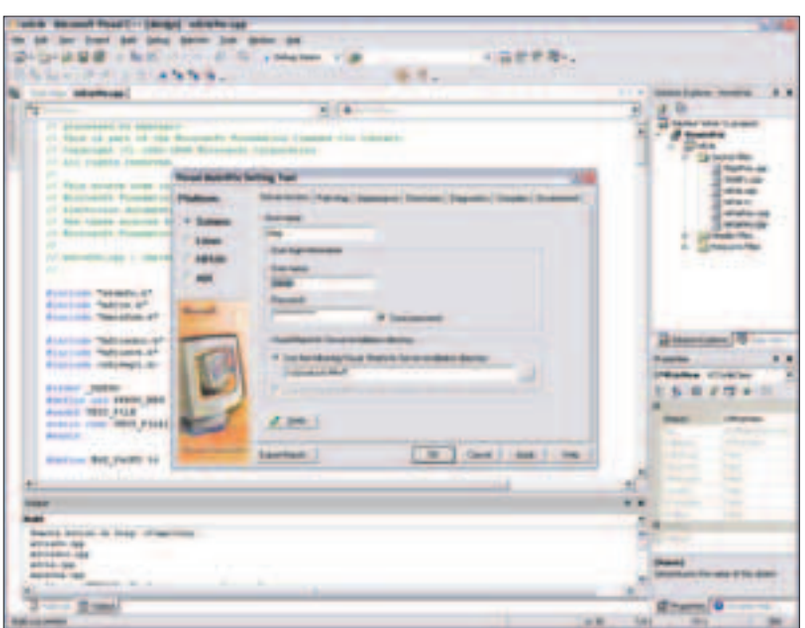
wxWindows - это бесплатная библиотека, практически полностью совместимая с MFC и работающая во всех *nix-системах, где есть GTK+, Motif или его бесплатный клон Lesstif. Единственное отличие заключается в том, что вместо префикса "C" здесь используется "wx", в результате чего CWnd превращается в wxWnd. Некоторые классы еще не реализованы (например, отсутствует CEditView), а когда они появятся - неизвестно. Это, конечно, неприятно, но и не смертельно. Без недостающих классов можно как-нибудь обойтись, заменив CEditVIew на wxTextCtrl, а операцию "перебивки" префиксов загнать в препроцессор или повесить на макрос. Самое главное - wxWindows прекрасно работает на Windows, а значит, один проект не распадется на два!

На сайте IBM есть замечательная статья по переносу MFC-приложений на wxWindows (см. врезку), а на сайте самой wxWindows еще немного материалов на эту тему. Судя по баннерам, проекту покровительствуют весьма впиятельные компании - VMWare и Helpware, поэтому за его дальнейшую судьбу можно не волноваться.

Множество полезных библиотек можно найти на www.sourceforge.net, например, библиотеку для работы с ini-файлами (не анализировать же ее с помощью Бизона!) - libini.lib. Все они

класс	MFC класс	wxWindows класс
Document	CDocument	wxDocument
View	CView	wxView
Edit view	CEditView	отсутствует
Template class	CMultiDocTemplate	wxDocTemplate
MDI parent frame	CMDIFrameWnd	wxDocMDIParentFrame
MDI child frame	CMDIChildWnd	wxDocMDIChildFrame
Document manager	отсутствует	wxDocManager

Соответствие основных классов между MFC и wxWindows



Так выглядит Visual WinMain

W W W

WINE
Популярный имитатор Windows, поддерживающий большое количество UNIX-платформ. Бесплатен: **www.winehq.org;**

WinX, он же Cedega
Коммерческий вариант WINE, ориентированный на игры и работающий преимущественно на LINUX-платформе: **www.transgaming.com;**

CodeWeavers
Коммерческий имитатор Windows, работающий только на Linux и ориентированный на запуск офисных приложений: **www.codeweavers.com;**

Visual MainWin
Плагин к Microsoft Visual Studio, упрощающий создание переносимого кода и позволяющий компилировать Windows-приложения под различные платформы. Здесь же лежит пара статей по переносу критических бизнес-приложений: **www.mainsoft.com/products/mainwin.html;**

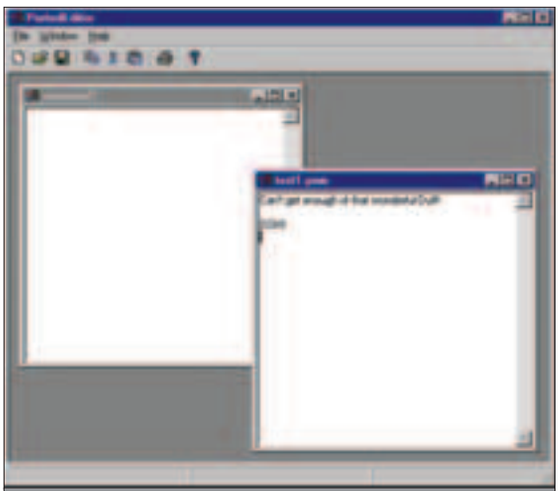
wxWindows
Кросс-платформенная библиотека, более или менее совместимая с MFC. Исходные тексты доступны, денег не просит: **www.wxwindows.org;**

LIBINT
Бесплатная библиотека для работы с INI-файлами на UNIX: **http://libini.sourceforge.net;**

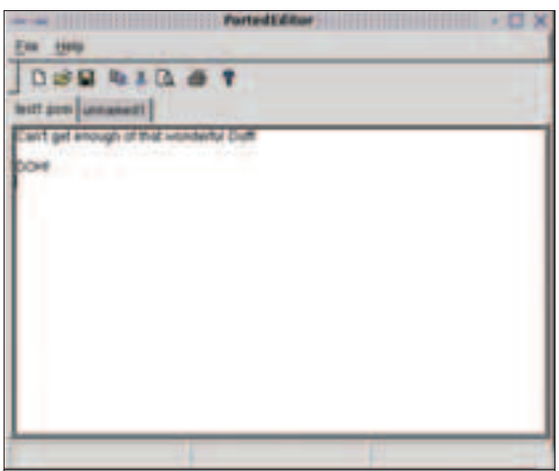
Free Pascal
Бесплатный кросс-платформенный компилятор Pascal'a с ограниченной поддержкой Delphi: **www.freepascal.org;**

Porting MFC applications to Linux
Толковая статья про перенос MFC-приложений в UNIX при помощи wxWindows: **www.106.ibm.com/developerworks/library/l-mfc;**

Иерархия классов MFC



Оригинальное MFC-приложение



Портированное на *nix приложение с помощью wxWindows

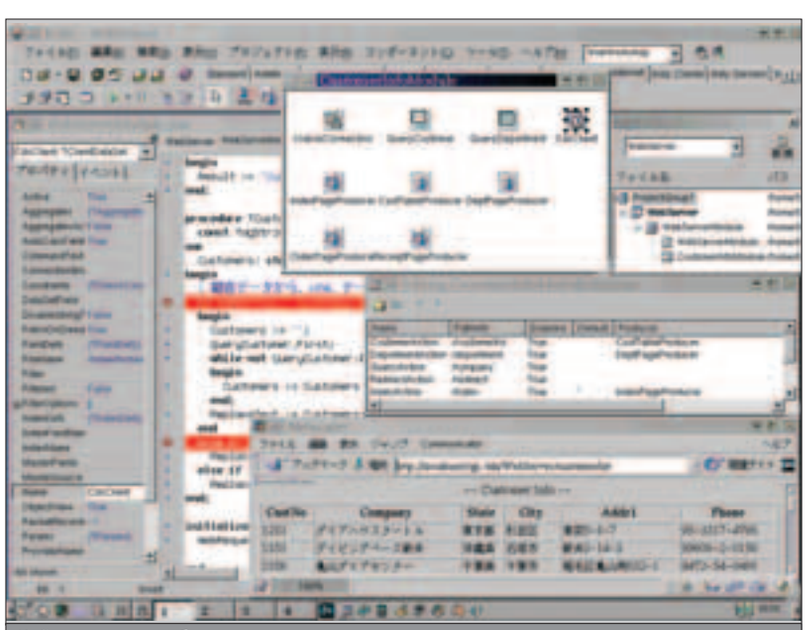
Иерархия классов wxWindows

wxWindows - это бесплатная библиотека, практически полностью совместимая с MFC

бесплатны, распространяются в исходных текстах и легко подключаются к любому проекту. Прежде чем писать код самому, обязательно посмотри, нет ли его в Сети уже. Скорее всего, он написан до тебя, так зачем же изобретать велосипед, когда есть готовые чертежи?

DELPHI + BUILDER + LINUX == KYLIX

Borland - великая фирма! Это она создала Turbo Pascal и Turbo Debugger (точнее не создала, а купила). Это она создала Turbo Vision и определила облик интегрированной среды разработки. Скажу честно: я не считаю Borland C++ хорошим компилятором (он как-то странно трактует ANSI-стандарт, да и оптимизирует плоховато), Builder я обхожу стороной, а от Delphi меня в прямом смысле тошнит. Но это - мои личные впечатления (солидарен - прим. Горл). »



Kylix в разгаре рабочего дня

Килик (древнегреч. kylix) - древнегреческий глиняный, реже металлический сосуд для питья вина: плоская чаша на подставке с двумя горизонтальными ручками. (из энциклопедического словаря)

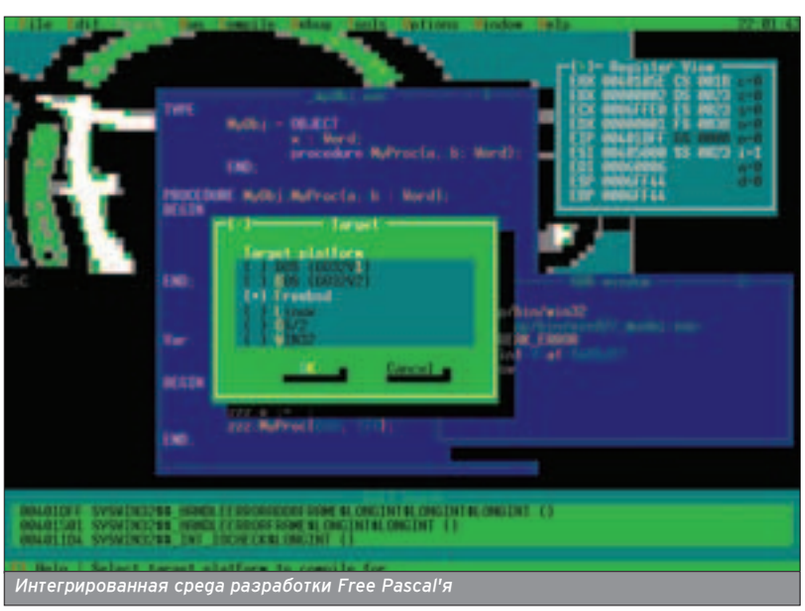
Мой любимый MS VC в *nix оказывается в очень затруднительном положении (перенос требует больших денежных вложений и телодвижений), а у Borland просто перекомпилируешь, и все!

Kylix - это Delphi и Builder для Linux, распространяющийся по лицензии GPL (то есть бесплатно) и включающий в себя интегрированную среду разработки (экранный редактор, интерактивный отладчик и т.п.) со всеми необходимыми библиотеками и слоями абстрагирования на борту. При условии, что программа не использует прямых вызовов win32 API, перенос не принесет никаких проблем (на самом деле все намного сложнее, и если это не чисто вычислительная задача типа бухгалтерии, без прямых вызов ей никак не обойтись - достаточно захотеть прочитать сектор с CD-ROM диска).

Win32	Linux
CreateProcess	fork()/execv()
TerminateProcess	kill
ExitProcess()	exit()
GetCommandLine	argv[]
GetCurrentProcessId	getpid
KillTimer	alarm(0)
SetEnvironmentVariable	putenv
GetEnvironmentVariable	getenv
GetExitCodeProcess	waitpid
Функции для работы с процессами	

Win32	Linux
_beginthread	pthread_attr_init
	pthread_attr_setstacksize
	pthread_create
_endthread	pthread_exit
TerminateThread	pthread_cancel
GetCurrentThreadId	pthread_self
TerminateThread((HANDLE *) threadId, 0);	pthread_cancel(threadId);
WaitForSingleObject ();	pthread_join();
_endthread();	pthread_exit(0);
Sleep (50)	struct timespec timeOut,remains;
	timeOut.tv_sec = 0;
	timeOut.tv_nsec = 500000000; /* 50 milliseconds */ nanosleep(&timeOut, &remains);
SleepEx (0,0)	sched_yield()
Функции для работы с потоками	

Win32	Linux
CreateFileMaping	mmap
OpenFileMapping	shmget
UnmapViewOfFile	munmap
	shmdt
MapViewOfFile	mmap
	shmat
UnmapViewOfFile(token->location); CloseHandle(token->hFileMapping);	munmap(token->location, token->nSize); close(token->nFileDes); remove(token->pFileName); free(token->pFileName);
Функции для работы с процессами	



Интегрированная среда разработки Free Pascal'я

> При условии, что программа не использует прямых вызовов win32 API, перенос не принесет никаких проблем.

А вот что действительно восхищает, так это Free Pascal (он же FPK Pascal) - бесплатный кросс-платформенный компилятор Pascal'а (с исходниками!), поддерживающий Intel x86, Motorola 680x0, PowerPC и работающий практически на любой операционной платформе: Linux, FreeBSD, NetBSD, MacOSX/Darwin, MacOS classic, DOS, Win32, OS/2, BeOS, Solaris, QNX и Amiga. Синтаксически и семантически Free Pascal полностью совместим с TP 7.0 и практически полностью - с Delphi версий 2 и 3. В дальнейшем планируется поддержка перекрытия функций и операторов. Kylix и рядом не валялся. На платформе Linux он король, а кто он за ее пределами?

Единственное, чего недостает Free Pascal - так это нормального IDE. Хотя на мой мыщьх'ый взгляд, тот IDE, который есть, гораздо удобнее MS VC и Delphi вместе взятых. Одно слово - консоль! При ближайшем рассмотрении выясняется другая замечательная вещь. Free Pascal не совсем компилятор, точнее, совсем не компилятор! Это - транслятор Pascal'я в C. Формально его можно считать компилятором переднего плана (Front-End Compiler), состыкованного с GCC. Отсюда и приличное качество оптимизации, и кросс-платформенность.

РУЧНОЙ ПЕРЕНОС, ИЛИ ОДИН НА ОДИН САМ С СОБОЮ

Смельчакам, отважившимся на самостоятельный перенос Windows-приложений, не обойтись без таблиц соответствий API-функций системным вызовам, которые приводятся ниже. Разуме-

W W W

- **C++ portability guide**
Шикарная карта рифов и отмелей с отметками всех несовместимостей различных компиляторов:
www.mozilla.org/hacking/portable-cpp.html;

- **UNIX Application Migration Guide**
Шикарное руководство по миграции из Windows в UNIX от Microsoft с многочисленными примерами. Подробно описаны все различия между этими системами, так что этот манускрипт работает в обе стороны:
www.willydev.net/descargas/prev/unix.pdf;

- **The Big Switch: Moving from Windows to Linux with Kylix 3**
Обзорная статья, описывающая перенос Delphi/Builder-приложений на Linux: **www-128.ibm.com/developerworks/db2/library/techarticle/0211swart/0211swart2.html;**

- **Migrating Win32 C/C++ applications to Linux on POWER**
Замечательная статья, посвященная "ручному" переносу приложений: **www-128.ibm.com/developerworks/eserver/articles/es-MigratingWin32toLinux.html;**

- **Using COM technologies on Unix platforms**
Как перенести COM-приложение на UNIX с минимальной головной болью:
www-128.ibm.com/developerworks/linux/library/l-com.html;

- **Реализация Win32 в среде ОС реального времени стандарта POSIX**
Перенос Windows-приложений на QNX; здесь же находится множество других интересных статей, посвященных этой великолепной, но мало известной операционной системе:
www.rts-ukraine.com/QNXArticles/willows_win32.htm;

- **A taste of Wine: Transition from Windows to Linux**
WINE как средство переноса приложений из Windows в UNIX:
www-128.ibm.com/developerworks/linux/library/l-wine/index.html;

- **OpenNT - путь к "открытому" NT?**
Обзор *nix-эмуляторов на Windows NT и Windows NT-эмуляторов на *nix: **www.osp.ru/os/1997/03/42.htm;**

- **Языки программирования через сто лет**
Какой язык выбрать для разработки долговременных приложений: **www.computerra.ru/hitech/35042;**

- **Портирование кода**
Подборка ссылок по портированию
www.opennet.ru/links/sml/50.shtml.

ется, это не все функции, а только самые популярные из них (полный список занял бы несколько увесистых томов, для транспортировки которых пришлось бы обзавестись грузовиком).

ЗАКЛЮЧЕНИЕ

Перенос Windows-приложений на *nix-системы намного проще, чем это кажется поначалу. К твоим услугам обширный инструментарий и огромное количество библиотек (в основном бесплатных). Сосредоточься на программном коде и забудь о пустяках - пусть ими занимается машина (см. эпиграф), но не откладывай это дело в долгий ящик и прекрати наконец игнорировать *nix-платформу. Ее популярность - факт. Так зачем терять рынок? Тем более что конкурировать здесь пока не с кем. В *nix до сих пор нет множества привычных Windows-приложений и утилит (систем распознавания текста, шестнадцатеричных редакторов и т.д.), поэтому даже плохонькая программа проглатывается публикой с энтузиазмом. Ты все еще ищешь, во что вонзить свои когти?

j1m (j1m@list.ru)

КАК *NIX-СИСТЕМЫ ПОТЕРЯЛИ ПОРТИРУЕМОСТЬ

ПРОГРАММИРУЕМ НА АССЕМБЛЕРЕ ПОД *NIX

Думаю, что ты уже хорошо овладел языком C под платформу *nix, и теперь у тебя появилось желание глубже изучить тонкости *nix-кодинга. Если это так, то тебе повезло - в этой статье ты узнаешь много полезного о программировании на ассемблере под Linux и BSD.

Вполне возможно, что пока ты читал заголовок статьи, у тебя возник вопрос: "А кому это надо?" Действительно, зачем во времена гигагерцовых процессоров, жестких дисков емкостью в несколько сот гигабайт и таких технологий, как Java и .NET, уметь программировать на ассемблере? Еще актуальнее этот вопрос для *nix. Все знают, что родным языком *nix'ов является C. Разработчики нашей любимой ОС сделали все, чтобы облегчить жизнь C-кодерам и чтобы бросить ассемблерщиков на произвол судьбы. Такой подход оправдан, потому как трудно найти задачу, которую можно было бы решить на асме, но нельзя на C. А тем, кто все-таки натыкался на подобные задачи, предоставлялся удобный механизм вставки ассемблерных инструкций в C-исходник. Ну и зачем нужен асм в такой невероятно портируемой ОС, как *nix? Чтобы ответить на этот вопрос, достаточно прочитать название журнала ;). Первое, что приходит на ум - вирусы. Вирус должен быть маленьким, быстрым, а его код максимально оптимизированным. Также знание ассемблера под конкретную платформу поможет в исследовании программ и дизассемблировании. Даже в открытой ОС это может понадобиться (для анализа вирусов ;)). Ну и третий довод в пользу асма - оптимизация (хотя в настоящее время это не так актуально).

> Первое, что бросается в глаза, когда начинаешь изучать синтаксис AT&T, так это его стройность и продуманность.

Описание системного вызова Open

AT&T VS. INTEL

Перед тем как перейти к изучению архитектурных особенностей *nix'ов, ознакомлю с особенностями ассемблера, применяемого в *nix-системах. Дело в том, что стандартный ассемблер, входящий в состав пакета Binutils и носящий имя "as", использует AT&T синтаксис. Что это значит? Компьютерному миру известно два синтаксиса: Intel-синтаксис, используемый во всех DOS- и Windows- ассемблерах, и AT&T-синтаксис, разработанный одноименной компанией и используемый в *nix-ассемблере "as". Чтобы двигаться дальше, необходимо освоить этот самый синтаксис AT&T, который имеет множество отличий от интеловского.

Первое, что бросается в глаза, когда начинаешь изучать синтаксис AT&T, так это его стройность и продуманность. В нем нет такого количества неоднозначностей, присутствующих в синтаксисе Intel. Таких, например, как неопределенная размерность операндов.

Итак, начнем. Основные отличия от Intel-синтаксиса:

❶. Порядок следования операндов противоположен привычному Intel-синтаксису, то есть сначала идет источник, а затем приемник.

❷. Названия регистров должны начинаться с символа '%', а непосредственно операнды - с символа '$'.

```
movl $10, %eax   # поместить в регистр eax число 10
```

❸. К командам, принимающим операнды, должен добавляться суффикс, указывающий размерность этих операндов: b - байт, w - слово, l - двойное слово, q - учетверенное слово, s - 32-битное число с плавающей точкой, l - 64-битное число с плаваю-

щей точкой, t - 80-битное число с плавающей точкой.

```
movb var, %al     # поместить в al байт по адресу var
movl %eax, %ebx   # поместить в ebx содержимое eax
```

❹. Адресация - это единственное, что в AT&T-синтаксисе сделано нелогично. В общем случае она выглядит так: смещение (базовый регистр, индексный регистр, множитель).

```
movl base_addr(%ebx,%edi,2),%eax
# mov eax,base_addr[ebx+edi*2]
```

Запятые сохраняются даже в случае отсутствия какого-либо элемента адресации.

❺. Директивы ассемблера начинаются с точки.

```
.string "Это строка"
```

❻. Специальный символ '.' - это ссылка на текущий адрес.

❼. Комментарии начинаются со знака '#' или заключаются в "/* */".

ДЕЛАЕМ ВСКРЫТИЕ ПИНГВИНУ

Что нужно знать об устройстве *nix-системы, чтобы писать программы на ассемблере? Как известно, всеми ресурсами ПК управляет операционная система. Она контролирует распределение оперативной памяти, управляет файловой системой и различными устройствами, такими как жесткий диск и звуковая карта. Поэтому любая программа, пожелавшая вывести на экран что-либо, создать файл, сменить текущий каталог или выполнить какое либо другое, внешнее по отношению к ней действие, должна просить об этом операционную систему. Запрос к операционной системе называется системным вызовом (Syscall). Существует множество различных системных вызовов (более 200), позволяющих открывать и читать файлы (Open, Read), создавать и уничтожать процессы (Fork, Kill), ходить по каталогам (Chdir) и делать еще множество разных полезных вещей. На С системный вызов выглядит как обращение к функции, например, чтобы открыть файл File.txt, достаточно одной строки:

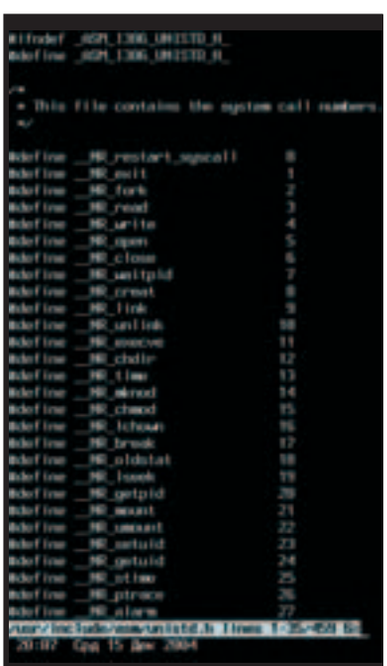
```
handle = open("file.txt", O_RDWR);
```

Как же будет выглядеть системный вызов на ассемблере? Это зависит от ядра ОС. В случае с Linux номер системного вызова помещается в регистр eax, а аргументы - в остальные регистры общего назначения. Далее необходимо выполнить запрос на прерывание с номером 80h. Приведу пример:

```
movl $5, %eax     # номер 5 (open - открыть файл) в eax
movl $file_name, %ebx
# адрес строки, содержащей имя файла в ebx
movl $02, %ecx
# 02 (числовое обозначение O_RDWR) в ecx
int $0x80     # системный вызов
```

В случае с BSD номер системного вызова опять же помещается в %eax, но аргументы кладутся в стек, затем необходимо выполнить все тот же int $0x80. Пример:

```
movl $5, %eax     # номер 5 (open - открыть файл) в eax
pushl $02     # 02 (числовое обозначение O_RDWR) в стек
pushl $file_name
# адрес строки, содержащей имя файла в стек
int $0x80     # системный вызов
```



Файл unistd.h в Linux

В обоих случаях дескриптор файла вернется в eax.

Почему же *nix-системы несовместимы друг с другом на низком уровне? На самом деле это только Linux несовместим со всеми остальными *nix'ами (Solaris, BSD и др.) :). Все остальные *nix'ы, вышедшие из оригинального UNIX (System 5), используют классическую конвенцию вызова, похожую на вызов функции в языке С. Linux же был написан с нуля (у Линуса даже стандарта POSIX на руках не было), Торвальдс получил большую свободу выбора и почему-то остановился на Fastcall конвенции вызова, свойственной продуктам от MS :).

В Linux номера системных вызовов можно посмотреть в файле /usr/include/asm/unistd.h, а в BSD - в файле /usr/include/sys/syscall.h. Описание самих системных вызовов можешь почитать в man'ах. Также советую заглянуть на страничку www.lxhp.in-berlin.de/lhpsyscal.html: там лежит описание работы с системными вызовами на низком уровне.

Помимо непосредственного обращения к ядру операционной системы существует и другой, более высокоуровневый и удобный способ общения с внешним миром - стандартная библиотека языка С (далее просто Libc). В языке С нет многих операторов, присутствующих в других языках программирования. Например, нет оператора печати строки и оператора выделения памяти. Вместо этого предоставляется богатый выбор функций, выполняющих аналогичные действия и входящих в библиотеку Libc (в настоящее время доминирует Glibc - GNU Libc). Хотя эта библиотека и разрабатывалась для взаимодействия с языком С, никто не мешает тебе воспользоваться ее услугами из других языков, в том числе и ассемблера. Для этого достаточно знать низкоуровневый формат вызова функций языка С, а он, кстати, достаточно прост. Необходимо перед вызовом функции занести параметры в стек. Вызов функции на С выглядит так:

```
puts(message);
```

на ассемблере примет такой вид:

```
pushl $message  # адрес строки в стек
call puts       # вызов функции
```

В случае с BSD единственное, что придется изменить в коде, - это поставить знак подчеркивания перед именами всех функций.

Хотя использование Libc и кажется простым, на самом деле это не так. Многие функции принимают в качестве аргументов или возвращают структуры, работать с которыми на асме не очень приятно. »

Статья рассчитана на тех, кто хотя бы поверхностно знаком с языком С и ассемблером. Поэтому, если ты не обладаешь необходимыми знаниями, тебе придется поднапрячься, чтобы понять все, о чем здесь рассказано.

В случае с Linux выбор метода обращения к функциям ОС не совсем понятен. Перед каждым системным вызовом приходится сохранять все регистры в стеке, затем производить системный вызов, а после этого восстанавливать прежнее содержимое регистров. С точки зрения производительности это накладно.

ПОЛЕЗНОЕ В СЕТИ

www.linuxassembly.org - официальный сайт Linux-ассемблерщиков
www.lowlevel.ru - информация по низкоуровневому программированию в Linux
www.lxhp.in-berlin.de/lhpsyscal.html - работа с системными вызовами на низком уровне
nasm.sourceforge.net - мультиплатформенный ассемблер с Intel-синтаксисом
ald.sourceforge.net - Assembly Language Debugger
www.linice.com - Попытка создать SoftIce под Linux
sources.redhat.com/gdb - The GNU Debugger

```
(gdb) b 1
Note: breakpoint 1 also set at pc 0x804838c.
Breakpoint 3 at 0x804838c: file glibc.s, line 1.
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/jim/coding/asm/glibc

Breakpoint 1, main () at glibc.s:12
12          pushl %ebp
(gdb) l
7       buf1: .space 256,0
8       buf2: .space 256,0
9
10      .text
11      main:
12          pushl %ebp
13          movl %esp,%ebp
14
15          pushl $mes1
16          call puts           # выводим сообщение (puts - вывести строку)
(gdb) s
13          movl %esp,%ebp
(gdb) s
main () at glibc.s:15
15          pushl $mes1
(gdb) s
16          call puts           # выводим сообщение (puts - вывести строку)
(gdb) p mes1
$2 = 1701603654
(gdb) _
```

Отладка программы в GDB

Описание всех функций Libc можно найти в третьей секции man-страниц.

Из сложившейся ситуации бинарной несовместимости BSD и Linux пользователи FreeBSD могут выйти благодаря Linux-эмуляции.

ЭКИПИРУЕМСЯ!

Теория - это, конечно, хорошо, и знание, как известно, - сила, но без практики далеко не уедешь. Для практики потребуются некоторые инструменты, такие как сам ассемблер, линковщик, и еще кое-что.

В первую очередь понадобится ассемблер, то есть транслятор, который будет переводить наши программы в машинные коды. Здесь у тебя есть два пути: не усложнять себе жизнь и использовать "as" с AT&T-синтаксисом, входящим в любой дистрибутив Linux и BSD или взять nasm с Intel-синтаксисом и мучиться с переписыванием примеров.

Еще одно важное орудие труда ассемблерщика - линковщик, который создает полноценные бинарники из объектных файлов, создаваемых ассемблером. Будем использовать стандартный линковщик из пакета Binutils под незамысловатым названием "ld".

Также тебе может понадобиться отладчик. В любой системе можно найти неплохой отладчик GDB, но он больше рассчитан на отладку C-программ. Существует также инструмент, специально предназначенный для ассемблерщиков, - Ald (Assembly Language Debugger). Любителям window`овского SoftIce советую посмотреть в сторону LinIce, который, кстати, представляет собой модуль ядра Linux (подробнее об отладчиках и отладке в *nix читай в этом Спеце).

Как же собирать программы? Способ создания исполняемых файлов зависит от того, используются ли в программе функции Libc. Для сборки программы, использующей только системные вызовы, проделай следующее:

Создать объектный файл:

```
$ as prog.s -o prog.o
```

И слинковать его:

```
$ ld prog.o -o prog
```

Так из исходника prog.s ты получишь бинарник prog.

Если же программа использует Libc, нужно выполнить только одну команду:

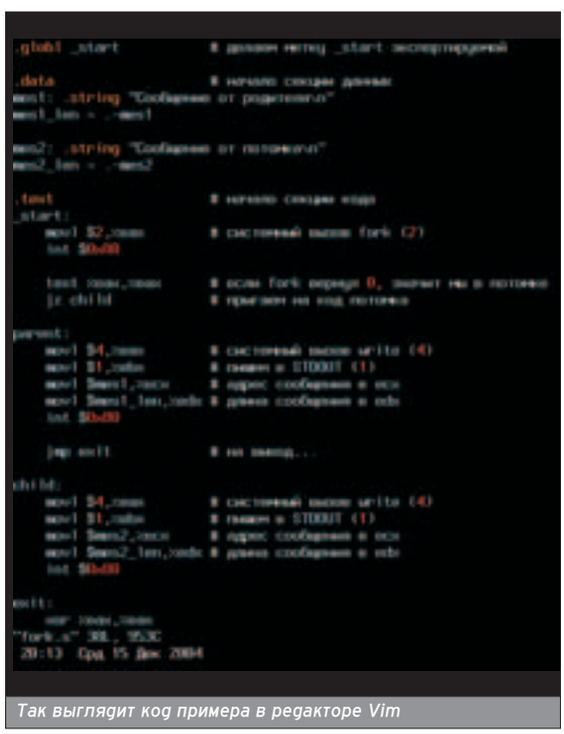
```
$ gcc prog.s -o prog
```

Компилятор языка C сам перенаправит программу ассемблеру и подключит необходимые объектные модули, иначе это пришлось бы делать вручную.

ПРАКТИЧЕСКИЕ ЗАНЯТИЯ

Для иллюстрации приведу пару примеров. Первый продемонстрирует работу с системными вызовами, а второй - с функциями Libc. Я не имею ничего против программистов, отдающих предпочтение BSD, но так как популярность Linux намного выше BSD, то и примеры будут под Linux.

```
.globl _start    # делаем метку _start экспортируемой

.data    # начало секции данных
mes1: .string "Сообщение от родителя\n"
mes1_len = .-mes1

mes2: .string "Сообщение от потомка\n"
mes2_len = .-mes2

.text    # начало секции кода
_start:
   movl $2,%eax    # системный вызов fork (2)
   int $0x80

   test %eax,%eax
# если fork вернул 0, значит мы в потомке
   jz child    # прыгаем на код потомка

parent:
   movl $4,%eax    # системный вызов write (4)
   movl $1,%ebx    # пишем в STDOUT (1)
   movl $mes1,%ecx    # адрес сообщения в ecx
   movl $mes1_len,%edx    # длина сообщения в edx
   int $0x80

   jmp exit    # на выход...

child:
   movl $4,%eax    # системный вызов write (4)
   movl $1,%ebx    # пишем в STDOUT (1)
   movl $mes2,%ecx    # адрес сообщения в ecx
   movl $mes2_len,%edx  # длина сообщения в edx
   int $0x80

exit:
   xor %eax,%eax
   inc %eax    # системный вызов exit (1)
```



Так выглядит код примера в редакторе Vim

> В любой системе можно найти неплохой отладчик GDB, но он больше рассчитан на отладку C-программ.

```
    xor %ebx,%ebx    # возвратим 0 (мол все нормально)
    int $0x80
```

Это классический пример разветвления процесса с помощью Fork. Процесс создает потомка и выводит сообщение о том, что сам он - родитель. Потомок же в свою очередь тоже идентифицирует себя. После вывода сообщения и родитель, и потомок завершаются.

Разберем часть исходного кода. В первой строке при помощи директивы .globl сообщаем о том, что метка _start является экспортируемой (глобальной). Метка _start должна присутствовать всегда, так как с этого адреса будет начинаться выполнение программы, а если не сделать ее глобальной, то линкер просто не увидит ее. Далее с помощью директивы .data объявляешь начало секции данных (в этой секции должны находится все статические данные, в нашем случае это строки). В этой секции по адресу mes1 находится строка. После нее - константа mes1_len, содержащая длину строки, которая вычисляется вычитанием адреса начала строки (метка mes1) из текущего адреса (директива '.'). Остальную часть секции данных занимает еще одна строка. После секции данных начинается секция кода (директива .text), в которой должны находиться все команды, выполняемые процессором. Остальная часть текста должна быть понятной. Описывать системные вызовы я не буду, так как они очень подробно описаны в документации aka man'ах.

А вот и пример с использованием Libc:

```
.globl main

.data
mes1: .string "File name:"
mes2: .string "New file name:"
buf1: .space 256,0
buf2: .space 256,0

.text
main:
    pushl %ebp    # создаем новый кадр стека
    movl %esp,%ebp

    pushl $mes1
    call puts
# выводим сообщение mes1 (puts - вывести строку)

    pushl $buf1
    call gets      # читаем имя файла в буфер buf1

    pushl $mes2
    call puts      # выводим сообщение mes2

    pushl $buf2
    call gets      # читаем новое имя файла

    add $16,%esp     # очищаем стек

    pushl $buf2     # новое имя файла в стек
    pushl $buf1     # старое имя файла в стек
    call rename     # переименование файла
    popl %ebx      # очищаем стек

    movl %ebp,%esp
# возвращаем стек в прежнее состояние
    popl %ebp

    ret        # выходим...
```

Перед тобой программа для переименования файлов. После запуска она задает пользователю вопрос о старом и новом имени файла, а затем переименовывает этот файл.

Как она работает? Как видно из исходника, программа представляет собой одну функцию, о чем говорит название метки (Main) и команда Ret в конце программы. Обрати внимание, что теперь экспортируем не метку _start, а метку Main. Почему? Вспомни С - в программе на этом языке обязательно должна присутствовать функция Main. Во время компиляции программа линкуется с некоторыми объектными файлами из Libc, в одном из которых находится (внимание!) метка _start, на которую и передается управление после запуска программы, библиотека выполняет некоторые (весьма полезные, кстати) действия и передает управление функции Main. Так как мы в своей программе пользуемся функциями Libc, то для того, чтобы получить управление, нам нужна функция Main. Как и в прошлой программе, в секции данных у нас находятся две строки и два буфера, заполненные нулями, каждый по 256 байт, созданные при помощи директивы .space. Заметь, что длину строк подсчитывать не надо: за тебя это сделает высокоуровневая функция Puts. Также не нужны переносы - символы переноса строки. Только одно замечание: Rename - это не функция, а системный вызов. Программа не вызывает его напрямую, а использует функцию-обертку, предоставляемую Libc.

Вспомни С - в программе на этом языке обязательно должна присутствовать функция Main.

```
localhost:~/coding/asm/example$ gcc glibc.s
/tmp/ccaaaYWJ.o(.text+0x13): In function `main':
: the `gets' function is dangerous and should not be used.
localhost:~/coding/asm/example$ l
итого 24k
drwx--x--x    2 jim      jim          4.0k Дек 15 20:14 ./
drwx------    3 jim      root         4.0k Дек 15 20:13 ../
-rwx--x--x    1 jim      jim           12k Дек 15 20:14 a.out*
-rw-------    1 jim      jim           697 Дек 15 20:13 glibc.s
localhost:~/coding/asm/example$ ./a.out
File name:
glibc.s
New file name:
ho-ho-ho.s
localhost:~/coding/asm/example$ l
итого 24k
drwx--x--x    2 jim      jim          4.0k Дек 15 20:15 ./
drwx------    3 jim      root         4.0k Дек 15 20:13 ../
-rwx--x--x    1 jim      jim           12k Дек 15 20:14 a.out*
-rw-------    1 jim      jim           697 Дек 15 20:13 ho-ho-ho.s
localhost:~/coding/asm/example$ ./a.out
File name:
ho-ho-ho.s
New file name:
glibc.s
localhost:~/coding/asm/example$ l
итого 24k
drwx--x--x    2 jim      jim          4.0k Дек 15 20:15 ./
drwx------    3 jim      root         4.0k Дек 15 20:13 ../
-rwx--x--x    1 jim      jim           12k Дек 15 20:14 a.out*
-rw-------    1 jim      jim           697 Дек 15 20:13 glibc.s
localhost:~/coding/asm/example$
```

Код из примера оказался работоспособным :)

По причине того, что в Linux аргументы системного вызова помещаются в регистры общего назначения, их количество ограничено шестью. Чтобы обойти это ограничение, нужно будет изловчиться.

ЗАКЛЮЧЕНИЕ

Надеюсь, эта статья помогла тебе получить представление о программировании на ассемблере под *nix. Если в ходе экспериментов у тебя возникнут какие-нибудь вопросы, ответы на которые ты не сможешь найти на страничках, указанных мною на врезке, то пиши мне, я постараюсь помочь тебе чем смогу.

Крис Касперски aka мыщъх

ОСОБЕННОСТИ НАЦИОНАЛЬНОЙ ОТЛАДКИ

ЗНАКОМИМСЯ С МЕХАНИЗМАМИ ОТЛАДКИ В *NIX

Первое знакомство в GDB (что-то вроде debug.com для MS-DOS, только мощнее) вызывает у поклонников Windows смесь разочарования с отвращением, а увесистая документация вгоняет в глубокое экзистенциональное уныние. Отовсюду торчат рычаги управления, но газа и руля нету. Не хватает только звериных шкур для дизайна и каменных топоров. Как юниксоиды выживают в агрессивной среде этого первобытного мира – загадка.

ОТЛАДКА В ИСТОРИЧЕСКОЙ ПЕРСПЕКТИВЕ

Несколько строчек исходного кода *nix еще помнят те древние времена, когда ничего похожего на интерактивную отладку не существовало и единственным средством борьбы с ошибками был аварийный дамп памяти. Программистам приходилось месяцами (!) ползать по вороху распечаток, собирая рассыпавшийся код в стройную картину.

Чуть позже появилась отладочная печать – операторы вывода, понатыканные в ключевых местах и распечатывающие содержимое важнейших переменных. Если происходил сбой, простыня распечаток (в просторечии "портянка") позволяла установить, чем занималась программа до этого и "кто ее так".

Отладочная печать сохранила свою актуальность и по сей день. В мире Windows она в основном используется в отладочных версиях программы и ликвидируется из финальной, что не есть хорошо: когда у конечных пользователей происходит сбой, в руках у них остается лишь аварийный дамп, на котором далеко не уедешь. Я согласен с тем, что отладочная печать кушает ресурсы и отнимает много времени. Вот почему в *nix так много систем управления протоколированием – от стандартного syslog до продвинутого Enterprise Event Logging (http://evlog.sourceforge.net). Они сокращают накладные расходы на вывод и журналирование, значительно увеличивая скорость выполнения программы.

Неправильно

```
#ifdef _DEBUG_
   fprintf(logfile, "a = %x, b = %x, c = %x\n", a, b, c);
#endif
```

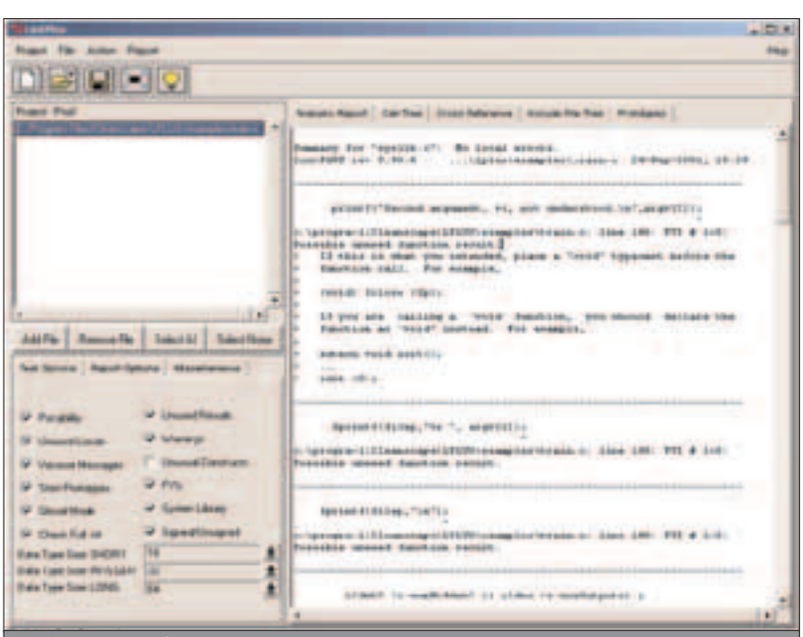
Правильно

```
if (_DEBUG_)
   fprintf(logfile, "a = %x, b = %x, c = %x\n", a, b, c);
```

Отладочная печать на 80% устраняет потребности в отладке, потому что отладчик используется в основном для определения того, как ведет себя программа в конкретном месте: выполняется ли условной переход, что возвращает функция, какие значения содержатся в переменных и т.д. Просто влепи сюда fprintf/syslog и посмотри на результат!

Человек – не слуга компьютеру! Это компьютер придуман для автоматизации человеческой деятельности (в мире Windows – наоборот!), поэтому *nix-системы на максимально возможном уровне "механизируют" поиск ошибок. Включи максимальный режим предупреждений компилятора или возьми автономные верификаторы кода (самый известный из которых LINT), и баги побегут из программы как мыщъх'и с тонущего корабля (Windows-компиляторы также могут генерировать сообщения об ошибках, по строгости не уступающие gcc, но большинство программистов пропускает их мимо ушей. Культура программирования, блин!).

Пошаговое выполнение программы и контрольные точки останова в *nix используются лишь в клинических случаях (например, при трепанации черепа), когда все остальные средства оказываются бессильными. Поклонникам Windows такой подход кажется несовременным, ущербным и жутко неудобным, но это все потому, что Windows-отладчики эффективно решают проблемы, которых в *nix-системах просто не возникает. Разница культуры программирования между Windows и *nix в действительности очень и очень значительная, поэтому прежде чем кидать камни в чужой огород, наведи порядок в своем. "Непривычное" еще не означает "неправильное". Точно такой же дискомфорт ощущает матерый юниксоид, очутившийся в Windows.



LINT в охоте на баги

PTRACE – ФУНДАМЕНТ ДЛЯ GDB

GDB – это системно-независимый кросс-платформенный отладчик. Как и большинство *nix-отладчиков, он основан на библиотеке PTrace, которая реализует низкоуровневые отладочные примитивы. Для отладки многопоточных процессов и параллельных приложений рекомендуется ис-

пользовать дополнительные библиотеки, например, CTrace (http://ctrace.sourceforge.net), а лучше – специализированные отладчики типа Total View (www.etnus.com), поскольку GDB с многопоточностью справляется не самым лучшим образом.

PTrace может: переводить процесс в состояние останова/возобновлять его выполнение, читать/записывать данные из/в адресное пространство отлаживаемого процесса, читать/записывать регистры ЦП. На i386 это – регистры общего назначения, сегментные регистры, регистры "сопроцессора" (включая SSE) и отладочные регистры семейства DRx (они нужны для организации аппаратных точек останова). В Linux еще можно манипулировать служебными структурами отлаживаемого процесса и отслеживать вызов системных функций. В "правильных" *nix-системах этого нет, и недостающую функциональность приходится реализовывать уже в отладчике.

PTRACE И ЕЕ КОМАНДЫ

В user-mode режиме доступна всего лишь одна функция – ptrace ((int _request, pid_t _pid, caddr_t _addr, int _data)), зато она делает все! При желании ты можешь за пару часов написать собственный мини-отладчик, специально заточенный под твою проблему.

Аргумент _request функции ptrace важнейший: он определяет, что мы будем делать. Заголовочные файлы в BSD и Linux используют различные определения, затрудняя перенос ptrace-приложений с одной платформы на другую. По умолчанию мы будем использовать определения из заголовочных файлов BSD.

PT_TRACE_ME (в Linux – PTRACE_TRACEME) - переводит текущий процесс в состояние останова. Обычно используется совместно с fork/exec, хотя встречаются также и самотрассирующиеся приложения. Для каждого из процессов вызов PT_TRACE_ME может быть сделан лишь однажды. Трассировать уже трассируемый процесс не получится (менее значительное следствие – процесс не может трассировать сам себя, сначала он должен расщепиться). На этом основано множество антиотладочных приемов, для преодоления которых приходится использовать отладчики, работающие в обход ptrace. Отлаживаемому процессу посылается сигнал, переводящий его в состояние останова, из которого он может быть выведен командами PT_CONTINUE или PT_STEP, вызванными из контекста родительского процесса. Функция wait задерживает управление материнского процесса до тех пор, пока отлаживаемый процесс не перейдет в состояние останова или не завершится (тогда она возвращает значение 1407). Остальные аргументы игнорируются.

PT_ATTACH (в Linux – PTRACE_ATTACH) - переводит в состояние останова уже запущенный процесс с заданным pid, при этот процесс-отладчик становится его "предком". Остальные аргументы игнорируются. Процесс должен иметь тот же самый UID, что и отлаживающий процесс, и не быть setuid/setduid процессом (или отлаживаться root'ом).

PT_DETACH (в Linux – PTRACE_DETACH) - прекращает отладку процесса с заданным pid (как по PT_ATTACH, так и по PT_TRACE_ME) и возобновляет его нормальное выполнение. Все остальные аргументы игнорируются.

PT_CONTINUE (в Linux – PTRACE_CONT) - возобновляет выполнение отлаживаемого процесса с заданным pid без разрыва связи с процессом-отладчиком. Если addr == 1 (в LINUX – 0), выполнение продолжается с места последнего останова, в противном случае – с указанного адреса. Аргумент _data задает номер сигнала, посылаемого отлаживаемому процессу (ноль – нет сигналов).

ПРИМЕР ИСПОЛЬЗОВАНИЯ PTRACE

```
// Подсчет кол-ва машинных команд в ls, для компиляции под Linux
// замени PT_TRACE_ME на PTRACE_TRACEME, а PT_STEP на PTRACE_SINGLESTEP

#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/ptrace.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <errno.h>

main()
{
 int pid; // pid отлаживаемого процесса
 int wait_val; // сюда wait записывает возвращаемое значение
 long long counter = 1; // счетчик трассируемых инструкций

 // расщепляем процесс на два, родитель будет отлаживать потомка
 // (обработка ошибок для наглядности опущена)
 switch (pid = fork())
 {
  case 0:   // дочерний процесс (его отлаживают)
     // папаша, ну-ка потрассируй меня!
     ptrace(PT_TRACE_ME, 0, 0, 0);
     // вызываем программу, которую надо оттрассировать
     // (для программ, упакованных "Шифрой", это не сработает)
     execl("/bin/ls", "ls", 0);
     break;
  default: // родительский процесс (он отлаживает)
     // ждем, пока отлаживаемый процесс не перейдет в состояние останова
     wait(&wait_val);
     // трассируем дочерний процесс, пока он не завершится
    while (WIFSTOPPED(wait_val) /* 1407 */)
    {
        // выполнить следующую машинную инструкцию и перейти в состояние останова
        if (ptrace(PT_STEP, pid, (caddr_t) 1, 0)) break;

        // ждем, пока отлаживаемый процесс не перейдет в состояние останова
        wait(&wait_val);

        // увеличиваем счетчик выполненных машинных инструкций на единицу
        counter++;
    }
 }
 // вывод количества выполненных машинных инструкций на экран
 printf("== %lld\n", counter);
}
```

"...отладка подобна охоте или рыбной ловле: те же эмоции, страсть и азарт. Долгое сидение в засаде в конце концов вознаграждается. Очередной невидимой миру победой..." - (c) Евгений Коцюба.

Добротно сверстанная документация на GDB (на русском языке): www.linux.org.ru/books/GNU/gdb/gdb-ru.pdf.

»

PT_STEP (в Linux – PTRACE_SINGLESTEP) - пошаговое выполнение процесса с заданным pid: выполнить следующую машинную инструкцию и перейти в состояние останова (под i386 это достигается взводом флага трассировки, хотя некоторые "хакерские" библиотеки используют аппаратные точки останова). BSD требует, чтобы аргумент addr был равен 1, Linux хочет видеть здесь 0. Остальные аргументы игнорируются.

PT_READ_I/PT_REEAD_D (в Linux – PTRACE_PEEKTEXT/PTRACE_PEEKDATA) - чтение машинного слова из кодовой области и области данных адресного пространства отлаживаемого процесса соответственно. На большинстве современных платформ обе команды совершенно эквивалентны. Функция ptrace принимает целевой addr и возвращает считанный результат.

PT_WRITE_I/PR_READ_D (в Linux – PTRACE_POKETEXT, PTRACE_POKEDATA) - запись машинного слова, переданного в _data, по адресу addr.

PT_GETREGS/PT_GETFPREGS/PT_GETDBREGS (в Linux – PTRACE_GETREGS, PTRACE_GETFPREGS, PTRACE_GETFPXREGS) - чтение регистров общего назначения, сегментных и отладочных регистров в область памяти процесса-отладчика, заданную указателем _addr. Это системно-зависимые команды, приемлемые только для i386 платформы. Описание регистровой структуры содержится в файле <machine/reg.h>.

PT_SETREGS/PT_SETFPREGS/PT_SETDBREGS (в Linux – PTRACE_SETREGS, PTRACE_SETFPREGS, PTRACE_SETFPXREGS) - установка значения регистров отлаживаемого процесса путем копирования содержимого региона памяти по указателю _addr.

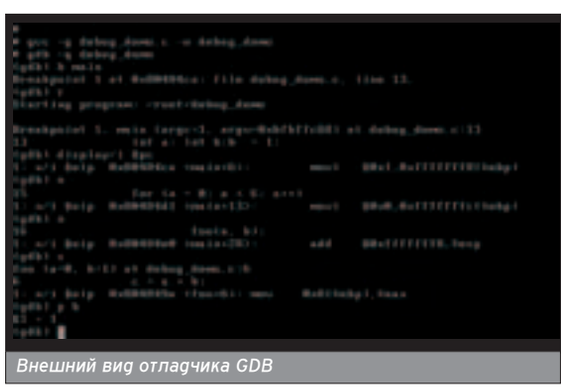
PT_KILL (в Linux – PTRACE_KILL) - посылает отлаживаемому процессу сигнал sigkill, который делает ему харакири.

Отличное руководство по внутреннему миру GDB (на английском языке). Очень помогает при доработке исходников: http://gnuarm.org/pdf/gdbint.pdf.

Статья про трассировку в Linux с примерами простейших трассировщиков (во FreeBSD иначе): http://gazette.linux.ru.net/lg81/sandeep.html.

КРАТКОЕ РУКОВОДСТВО ПО GDB

GDB – это консольное приложение, выполненное в классическом духе командной строки. И хотя за время своего существования GDB успел обрасти ворохом красивых графических морд, интерактивная отладка в стиле TD в мире *nix не очень популярна. Как правило, это удел эмигрантов с Windows-платформы, сознание которых необратимо искалечено идеологией "окон". Грубо говоря, если TD – слесарный инструмент, то GDB – токарный станок с программным управлением. Когда-нибудь ты полюбишь его...



Внешний вид отладчика GDB

Для отладки на уровне исходных текстов программа должна быть откомпилирована с отладочной информацией. В gcc за это отвечает ключ "-g". Если отладочная информация недоступна, GDB будет отлаживать программу на уровне дизассемблерных команд.

Обычно имя отлаживаемого файла передается в командной строке (gdb filename). Для отладки активного процесса укажи в командной строке его ID, а для подключения коды (core dump) воспользуйся ключом "-core==corename". Все три параметра можно загружать одновременно, попеременно переключаясь между ними командной target. Target exec переключается на отлаживаемый файл, target child - на приаттаченный процесс, а target core - на дамп коры. Необязательный ключ "-q" подавляет вывод копирайта.

Загрузив программу в отладчик, нужно установить точку останова. Для этого служит команда break (она же "b"). b main устанавливает точку останова на функцию main языка C, а b _start – на точку входа в ELF-файл (впрочем, в некоторых файлах она называется по-другому). Можно установить точку останова и на произвольный адрес: b *0x8048424 или b *$eax. Регистры пишутся маленькими буквами и предваряются знаком доллара. GDB понимает два "общесистемных" регистра: $pc (указатель команд) и $sp (стековый указатель). Только помни, что непосредственно после загрузки программы в отладчик никаких регистров у нее еще нет, и они появляются только после запуска отлаживаемого процесса на выполнение (команда run, она же "r").

Отладчик самостоятельно решает, какую точку останова установить – программную или аппаратную, и лучше ему не препятствовать (команда принудительной установки аппаратной точки останова (hbreak) работает не на всех версиях отладчика; в моей она не работает точно). Точки останова на данные в GDB называются "точками наблюдения" – watch point. Watch addr вызывает отладчик вся-

ПОДДЕРЖКА МНОГОПОТОЧНОСТИ В GDB

Определить, поддерживает ли твоя версия GDB многопоточность, можно при помощи команды info thread (вывод сведений о потоках), а для переключений между потоками используй thread N.

Если поддержка многопоточности отсутствует, обнови GDB до версии 5x или установи специальный патч, поставляемый вместе с твоим клоном UNIX или распространяемый отдельно от него.

Внешний вид отладчика Total View

Отладка многопоточных приложений не поддерживается:

```
(gdb) info threads
(gdb)
```

Отладка многопоточных приложений поддерживается:

```
info threads
  4 Thread 2051 (LWP 29448) RunEuler (IpvParam=0x80a67ac) at eu_kern.cpp:633
  3 Thread 1026 (LWP 29443) 0x4020ef14 in __libc_read () from /lib/libc.so.6
* 2 Thread 2049 (LWP 29442) 0x40214260 in __poll (fds=0x80e0380, nfds=1, timeout=2000)
  1 Thread 1024 (LWP 29441)  0x4017caea in __sigsuspend (set=0xbffff11c)
(gdb) thread 4
```

ТРАССИРОВКА СИСТЕМНЫХ ФУНКЦИЙ

Перехват системных функций – это настоящее окно во внутренний мир подопытной программы, показывающее имена вызываемых функций, их аргументы и коды возврата. Отсутствие "лишних" проверок на ошибки - болезнь всех начинающих программистов, и отладчик – не самое лучшее средство их поиска. Воспользуйся одной из штатных утилит truss/ktrace или возьми любой бесплатный/коммерческий анализатор.

кий раз, когда содержимое addr изменяется, а awatch addr – при чтении/записи в addr. Команда rwatch addr реагирует только чтение, но работает не во всех версиях отладчика. Просмотреть список установленных точек останова/наблюдения можно командой info break. Команда clear удаляет все точки останова, clear addr – все точки останова, установленные на данную функцию/адрес/номер строки. Команды enable/disable позволяют временно включать/отключать точки останова. Точки останова поддерживают развитый синтаксис условных команд, описание которого можно найти в документации. Команда continue ("c") возобновляет выполнение программы, прерванное точкой останова.

Команда next N ("n N")выполняет N-следующих строк кода без входа, а step N ("s N") - со входом во вложенные функции. Если N не задано по умолчанию, выполняется одна строка. Команды nexti/stepi делают то же самое, но работают не со сроками исходного текста, а с машинными командами. Обычно они используются совместно с командой display/i $pc ("x/i $pc"), предписывающей отладчику отображать текущую машинную команду. Ее достаточно вызывать один раз за сеанс.

Команда jump addr передает управление в произвольную точку программы, а call addr/fname – вызывает функцию fname с аргументами! Этого нету даже в SoftIce! А как часто оно требуется! Другие полезные команды: finish – продолжать выполнение до выхода из текущей функции (соответствует команде soft-ice "P RET"), until addr ("u addr") – продолжать выполнение, пока указанное место не будет достигнуто, при запуске без аргументов - остановить выполнение при достижении следующей команды (актуально для циклов!), return – немедленное возвращение в дочернюю функцию.

Команда print выражение ("p выражение") выводит значение выражения (например, "p 1+2"), содержимое переменной ("p my_var"), содержимое регистра ("p $eax") или ячейку памяти ("p *0x8048424", "p *$eax"). Если нужно вывести несколько ячеек – воспользуйся командой x/Nh addr, где N – количество выводимых ячеек. Ставить символ звездочки перед адресом в этом случае не нужно. Команда info registers ("i r") выводит значение всех доступных регистров. Модификация содержимого ячеек памяти/регистров осуществляется командой set. Например, set $eax = 0 записывает в регистр eax ноль. set var my_var = $ecx присваивает переменной my_var значение регистра ecx, а set {unsigned char*}0x8048424=0xCC записывает по байтовому адресу 0x8048424 число 0xCC. disassemble _addr_from _addr_to выдает содержимое памяти в виде дизассемблерного листинга, формат представления которого определяется следующей командой: set disassembly-flavor.

Команды info frame, info args, info local отображают содержимое текущего фрейма стека, аргументы функции и локальные переменные. Для переключения на фрейм материнских функций служит команда frame N. Команда backtrace ("bt") делает то же самое, что и call stack в Windows-отладчиках. При исследовании дампов коры она незаменима.

Короче говоря, приблизительный сеанс работы с GDB выглядит так: грузим программу в отладчик, даем b main (а если не сработает, то b _start), затем "r", после чего отлаживаем программу по шагам: "n"/"s", при желании задав "x/i $pc", чтобы GDB показывал, что выполняется в данный момент. Выходим из отладчика командой quit ("q"). Описание остальных команд – в документации. Теперь, по крайней мере, ты не заблудишься в ней.

ЗАКЛЮЧЕНИЕ

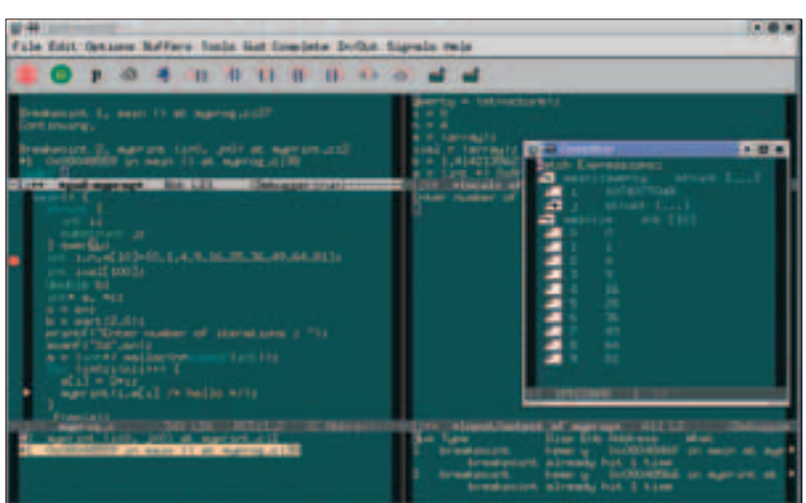
Сравнение *nix-отладчиков с Windows-дебаггерами показывает значительное отставание последних и их непрофессиональную направленность. Трехмерные кнопки, масштабируемые иконки, всплывающие менюшки – все это, конечно, очень красиво, но жать <F10> до потери пульса лениво. В GDB проще написать макрос или использовать уже готовый (благо все, что только можно было запрограммировать, здесь сделали уже до тебя и меня).

Отладочные средства в *nix мощны и разнообразны (свет клином не сошелся на GDB), и единственное, чего ей недостает, – так это нормального ядерного отладчика системного уровня, ориентированного на работу с двоичными файлами без символьной информации и исходных тестов. Тяжелое детство и скитание по множеству платформ наложило на *nix тяжелый отпечаток и ничем не исстребимое стремление к переносимости и кросс-платформенности. Какое там хакерство в таких условиях! Впрочем, доступность исходных текстов делает эту проблему неактуальной.

Использование библиотеки CTrace для отладки многопоточных программ (на английском языке): www.linux-mag.com/2004-04/code_01.html.

Исследование и отладка ELF-файлов на i386-платформе без исходных текстов: www.sstic.org/SSTIC03/articles/SSTIC03-Vanegue_Roy-Reverse_Intel_ELF.pdf.

Отладчик DDD – графический интерфейс к GDB



Еще один графический интерфейс к GDB

Крис Касперски aka мыщъх

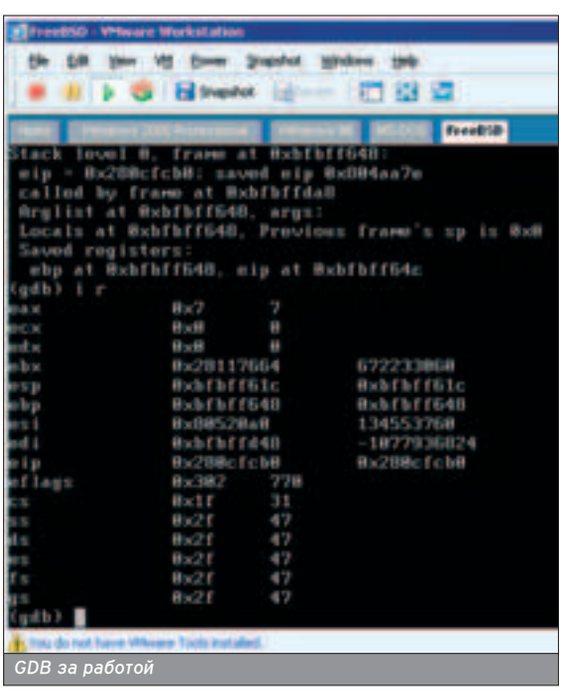
НЕСЕТЕВАЯ ЗАЩИТА

МЕТОДОЛОГИЯ ЗАЩИТЫ СОФТА В *NIX

Качество защитных механизмов в *nix все еще остается на очень низком уровне, и с Windows ей не соперничать. Впрочем, качество хакерского инструментария под *nix еще хуже, так что даже плохенькая защита для взломщика ПО станет большой проблемой и дикой головной болью. В этой статье - кратко об этой проблеме и о наиболее популярных методах противодействия отладчикам и дизассемблерам в семействе *nix-систем.

Программное обеспечение под *nix далеко не всегда бесплатно, и коммерческий софт успешно конкурирует с OpenSource-проектами, многие из которых, кстати, распространяются за деньги ("свободное ПО" еще не означает "бесплатное ПО"). Это и научные приложения, моделирующие движения звезд в галактиках, и корпоративные пакеты для работы с трехмерной графикой, и серверное обеспечение, и программные комплексы для управления производством, и т.д. и т.п. Все это не имеет никакого отношения ни к ПК, ни к "пиратству". Исследовательские институты и корпорации слишком дорожат своей репутацией, чтобы идти на открытый грабеж.

Именно поэтому в мире *nix так мало способов защиты от несанкционированного копирования. Хотя Linux в этом отношении - исключение. Ориентированная на использование на домашних и офисных компьютерах, она идет по тропе варварского рынка (он же - "массовый рынок"), на котором обитают хакеры, пираты и продвинутые пользователи, способные постоять за свои права наскоро скачав из Сети свежий crack. Без достойной защиты никуда! Без достойной защиты твоя программа вообще не будет продаваться.



GDB за работой

РАЗВЕДКА ПЕРЕД БОЕМ

Для хакера ПО *nix-системы - не очень интересное место. Достойного инструментария здесь нет, и не будет даже в скором времени. Взламывать софт приходится голыми руками (немного помогая головой). Больше всего удручает отсутствие полноценного отладчика, если не SoftIce'a, то хотя бы OllyDbg. Мелочи наподобие дамперов памяти, разных патчеров, автоматических распаковщиков тоже придется писать самостоятельно, поскольку живых представителей этой фауны вряд ли удастся обнаружить в Сети. Повсюду только бесконечные кладбища заброшенных проектов.

Будем надеяться, что через несколько лет ситуация измениться (как известно, спрос рождает предложение), а пока ограничимся кратким обзором существующего софта, полезного при взломе приложений в *nix.

ОТЛАДЧИКИ

GDB - кросс-платформенный source-level отладчик, основанный на библиотеке Ptrace (см. man ptrcae) и ориентированный преимущественно на отладку приложений с исходными текстами. Для взлома подходит плохо, если подходит вообще. Поддерживает аппаратные точки останова на исполнение (однако при запуске из-под VMWare они не срабатывают, а на голом железе я его не гонял), но не тянет чтение/запись памяти. Не может брякать и модифицировать совместно используемую память (то есть ls с его помощью ты вряд ли отладишь!). Поиск в памяти отсутствует как таковой. Отказывается загружать файл с искаженной структурой или с отрезанной Section table. Внешне представляет собой консольное приложение со сложной системой команд, полное описание которых занимает порядка трехсот страниц убористого текста. При желании к отладчику можно прикрутить графическую оболочку (благо недостатка в них нет), однако красивым интерфейсом кривое ядро не исправишь. За время своего существования GDB успел обрасти густой шерстью антиотладочных приемов, которые в основном до сих пор актуальны. GDB бесплатен, распространяется по лицензии GNU (отсюда и название - Gnu DeBugger), входит в комплект поставки большинства *nix-систем и к тому же позволяет патчить исполняемый файл не выходя из отладчика.

Краткое руководство для начинающих: чтобы брякнуться на точке входа, необходимо предварительно определить ее адрес, для чего пригодится штатная утилита Objdump (только для незащищенных файлов!) или biew/IDA: objdump file_name -f. Потом, загрузив отлаживаемую программу в GDB (gdb -q file_name), дать команду break *0xXXXXXXXX, где "0xX" - стартовый адрес, а затем Run для ее запуска на выполнение. Если все прошло успешно, GDB тут же остановится и передаст тебе бразды правления. Если же нет, открой файл в Biew и внедри в Entry point точку останова (код CCh), предварительно сохранив (в голове) оригинальное содержимое, перезапусти отладчик, а после достижения точки останова восстанови

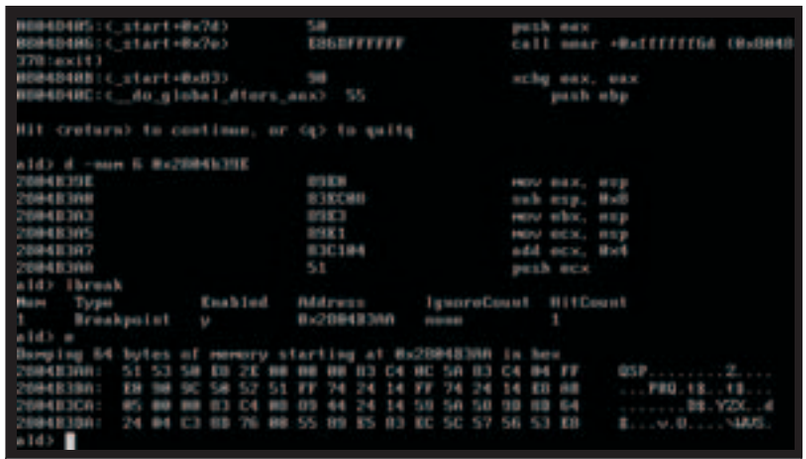
Чтобы брякнуться на точке входа, необходимо предварительно определить ее адрес.

ее содержимое (set {char} *0xXXXXXXXX = YY).

ALD: Assemble Language Debugger (http://ald.sourceforge.net) – пронырливый source-level application-debugger с минимумом рычагов управления, ориентированный на отладку ассемблерных текстов и двоичных файлов. Основан на библиотеке Ptrace со всеми вытекающими последствиями. В настоящее время работает только на x86-платформе, успешно компилируясь под следующие операционные системы: Linux, FreeBSD, NetBSD, OpenBSD. Поддерживает точки останова на выполнение, пошаговую/покомандную трассировку, просмотр/редактирование дампа, простор/изменение регистров, а также содержит простенький дизассемблер. Довольно аскетичный набор для взлома программ! Достопочтенный debug.com для MS-DOS и тот побогаче. Зато ALD бесплатен, распространяется в исходных текстах и грузит файлы без Section table. Для обучения взлому он вполне подойдет, но на звание основного хакерского инструмента, увы, не тянет.

THE DUDE (http://the-dude.sourceforge.net) – интересный Source-level отладчик, работающий в обход Ptrace и успешно работающий там, где gdb/ald уже не справляются. К сожалению, работает только под Linux - поклонникам остальных операционных систем с этой утилитой поработать не удастся. Архитектурно состоит из трех основных частей: модуля ядра the_dude.o, реализующего низкоуровневые отладочные функции, спрягающей библиотечной обертки вокруг него – libduderino.so и внешнего пользовательского интерфейса – ddbg. Собственно говоря, пользовательский интерфейс лучше переписать сразу. Отладчик бесплатен, но для его скачивания требуется предварительная регистрация на www.sourcefogre.net

LINICE (www.linice.com) – SoftIce под Linux. Чрезвычайно мощный отладчик ядерного уровня, ориентированный на работу с двоичными файлами без исходников. Основной инструмент любого хакера, работающего под Linux. В настоящее время работает только на ядре версии 2.4 (и вроде бы на 2.2 тоже) и отваливается с ошибкой в файле Iceface.c при компиляции под все остальные. Добавляет устройство /dev/ice, чем легко выдает свое присутствие в системе (впрочем, благодаря наличию исходных текстов это не будет серьезной проблемой). Всплывает при нажатии <CTRL>+Q, причем USB-клавиатура пока не поддерживается, так что для взлома придется использовать старую PS/2. Загрузчика нет и не предвидится, поэтому единственным способом отладки остается внедрение машинной команды INT 03 (опкод CCh) в точку входа с последующим ручным восстановлением оригинального содержимого.



Внешний вид отладчика ALD

PICE (http://pice.sourceforge.net) - экспериментальный ядерный отладчик для Linux, работающий только в консольном режиме и реализующий, к сожалению, минимум функций. Тем не менее, и он может сгодиться на что-нибудь.

The x86 Emulator plugin for IDA Pro (http://ida-x86emu.sourceforge.net) – эмулирующий отладчик, конструктивно выполненный в виде плагина для IDA Pro и распространяющийся в исходных текстах без предкомпиляции (а это значит, что кроме самой IDA Pro еще понадобиться и SDK, найти которой намного труднее). Основное достоинство эмулятора в том, что он позволяет выполнять произвольные куски кода на виртуальном процессоре. Например, передавать управление процедуре проверки серийного номера/пароля минуя остальной код. Такая техника совмещает лучшие черты статического и динамического анализа, значительно упрощая взлом законовыристых защит.

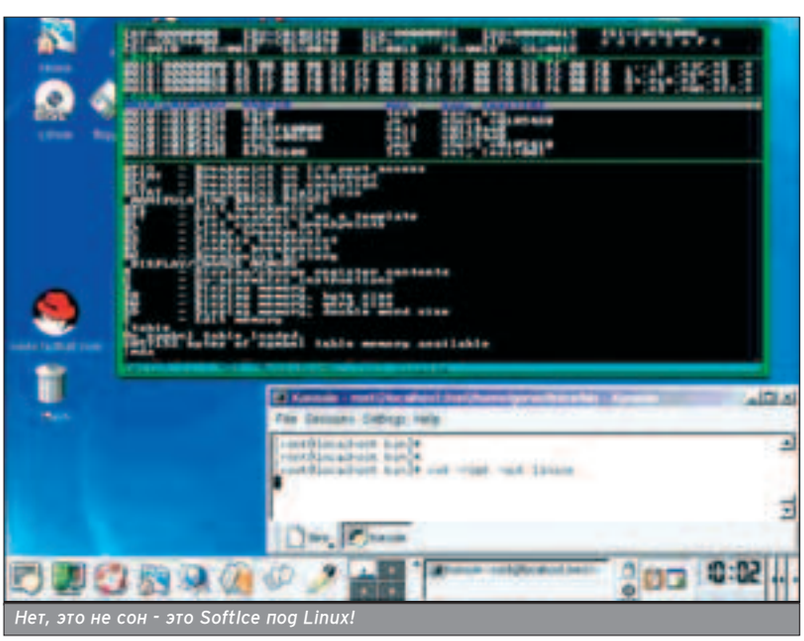
ДИЗАССЕМБЛЕРЫ

IDA Pro (www.idapro.com) – лучший дизассемблер всех времен и народов, теперь доступен и под Linux! Поклонники же FreeBSD и остальных операционных систем могут довольствоваться консольной Windows-версией, запущенной под эмулятором, или работать с ней непосредственно из-под MS-DOS, OS/2, Windows. До недавнего времени IDA Pro отказывалась дизассемблировать файлы без Section table, однако в последних версиях этот недостаток был устранен. Отсутствие приличных отладчиков под *nix превращает IDA Pro в основной инструмент изучения той или иной защиты.

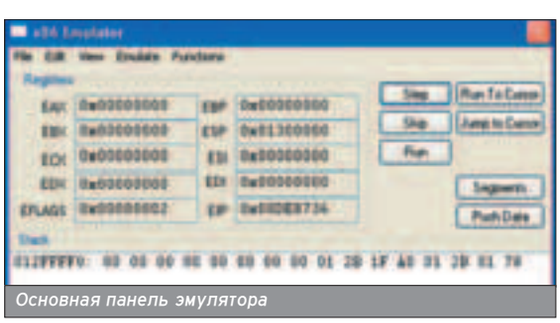
Objdump – аналог Dumpbin для ELF-файлов с простеньким дизассемблером внутри. Требует обязательного наличия Section table, не переваривает искаженных полей, с упакованными файлами не справляется. Тем не менее, при отсутствии IDA Pro сгодится и она.

С точки зрения хакера ПО *nix-системы – не очень интересное место.

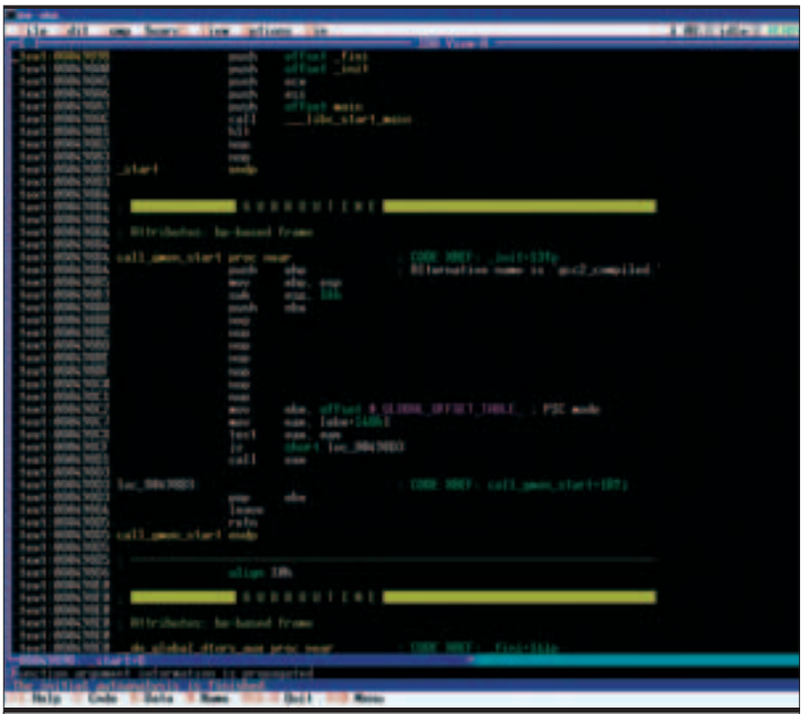
В *nix-системах содержимое памяти каждого из процессоров представлено в виде набора файлов, расположенных в каталоге /proc.



Нет, это не сон - это SoftIce под Linux!



Основная панель эмулятора



Консольная версия IDA Pro

ШПИОНЫ

Truss – полезная утилита, штатным образом входящая в комплект поставки большинства *nix-систем. Отслеживает системные вызовы (они же – Syscalls) и сигналы (Signals), совершаемые подопытной программой с прикладного уровня, что позволяет сказать многое о внутреннем мире защитного механизма.

Ktrace – еще одна утилита из штатного комплекта поставки. Отслеживает системные вызовы, Namei translation (синтаксический разбор имен), операции ввода-вывода, сигналы, userland-трассировку и переключение контекстов, совершаемых подопытной программой с ядерного уровня. Короче говоря, Ktrace представляет собой улучшенный вариант Truss, но, в отличие от последней, выдает отчет не в текстовой, а двоичной форме, и для генерации отчетов необходимо будет воспользоваться утилитой Kdump.

ШЕСТНАДЦАТЕРИЧНЫЕ РЕДАКТОРЫ

BIEW (http://belnet.dl.sourceforge.net/sourceforge/biew/biew562.tar.bz2) – HEX-редактор, дизассемблер, криптор и инспектор ELF-формата в одном флаконе. Встроенный ассемблер отсутствует, поэтому модифицировать программу приходится непосредственно в машинном коде, что напрягает. Но выбора все равно нет (разве что дописать ассемблер самостоятельно).

ДАМПЕРЫ

В *nix-системах содержимое памяти каждого из процессоров представлено в виде набора файлов, расположенных в каталоге /proc. Здесь же хранится контекст регистров и все остальное. Однако дамп памяти – это еще не готовый ELF-файл, и к непосредственному употреблению он не пригоден. Тем не менее, дизассемблировать его "сырой" образ вполне возможно.

АВТОМАТИЗИРОВАННЫЕ СРЕДСТВА ЗАЩИТЫ

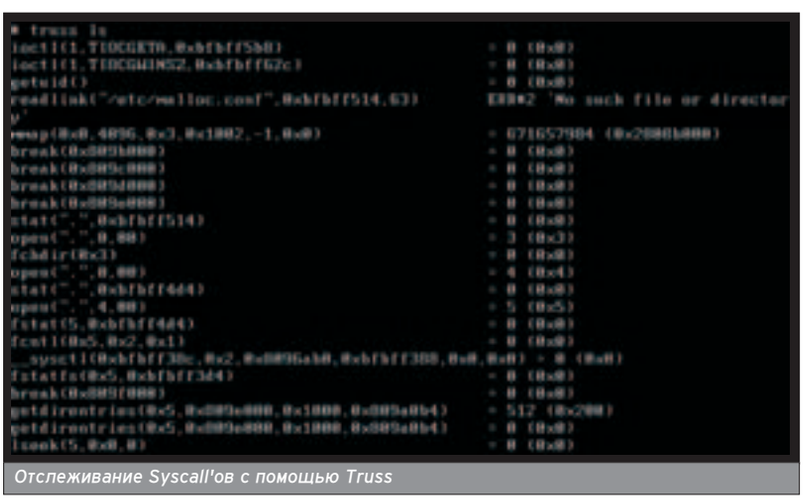
Упаковщики исполняемых файлов используются не только для уменьшения размеров программы, но и для затруднения ее взлома. Под Windows такая мера никого не остановит, а вот *nix – другое дело! Автоматических распаковщиков нет, дампер и не ночевали, отлаживать нечем (кроме Linux-систем - там-то есть достойный отладчик). Просто пропускаешь файл через упаковщик, и тогда его никто не расковыряет. То есть расковырять, конечно, смогут, но для этого хакеру понадобятся весьма серьезные мотивы к этому, чего у него обычно нет.

Минус всех упаковщиков в том, что они серьезно снижают мобильность защищенной программы (в особенности если содержат системно-зависимые антиотладочные приемы), к тому же все известные мне упаковщики нацелены исключительно на Linux и не работают под FreeBSD и другие UNIX-клоны, хотя в написании такого упаковщика нет ничего невозможного.

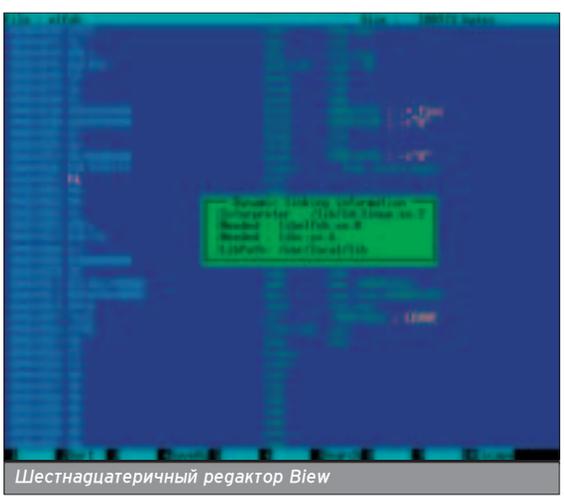
Shiva (www.securereality.com.au) – самый мощный упаковщик из всех имеющихся, хотя и основан на морально устаревших идеях, известных Windows-программистам с незапамятных времен. Реализует многослойную модель шифровки по типу "лука" (onion's layer) или "матрешки", использует полиморфный движок, нашпигованный множеством антиотладочных и антидизассемблерных приемов, противодействует Gdb и другим отладчикам, работающим через Ptrace, успешно борется с strace/ltrace/fenris, а также предотвращает снятие скальпа (то есть дампа) программы через /proc. Подробности на Blackhat: www.blackhat.com/presentations/bh-federal-03/bh-federal-03-eagle/bh-fed-03-eagle.pdf. Вопреки распространенному мнению о несокрушимости Shiva, для опытного хакера она не преграда. К тому же агрессивная природа упаковщика приводит к многочисленным проблемам, например, перестает работать Fork. Тем не менее, появление Shiva – большой шаг впе-

Большинство антиотладочных приемов по своей природе системно-зависимы и препятствуют переносу защищенной программы на другие платформы.

Функцию ptrace нельзя вызывать дважды – попытка трассировки уже трассируемого процесса порождает ошибку.



Отслеживание Syscall'ов с помощью Truss



Шестнадцатеричный редактор Biew

peg, и для защиты от начинающих взломщиков это лучший выбор!

Burneye (http://packetstormsecurity.nl/groups/teso/burneye-1.0.1-src.tar.bz2) – популярный, но не слишком стойкий упаковщик/протектор. Уже давно взломан, и руководство по его преодолению в Сети не найдет только ленивый. Вот только некоторые из них: www.securitylab.ru/tools/32046.html, www.activalink.net/index.php/BurnEye Encrypted Binary Analisis, www.incidents.org/papers/ssh_exploit.pdf. Использует крайне примитивный механизм определения отладчика – просто подает сигнал 5 (Trace/breakpoint trap), в отсутствие GDB или чего-то очень на него похожего передающий управление на специальную процедуру, увеличивающую значение "секретной" ячейки памяти на единицу, а в присутствие – выпетающий в отладчик. При наличии "правильного" отладчика, работающего в обход Ptrace, наподобие THE DUDE или LINICE, ломается элементарно, хотя и не так быстро, как хотелось бы (приходится продираться через тонны запутанного кода, напоминающего мычание коровы, погулявшей на маковом поле). Для защиты от невъедливого хакера Bruneye вполне подходит, а большего нам чаще всего и не надо!

624 (http://sed.free.fr/624) – малоизвестный простенький упаковщик. Работает шесть дней в неделю по 24 часа, а в воскресенье отдыхает. Шутка! Но добавить его к своей коллекции все-таки стоит.

Upx (http://upx.sourceforge.net) – легендарный кросс-платформенный упаковщик, работающий на множестве платформ от Atari go Linux. Никак не препятствует отладке и, что хуже всего, содержит встроенный распаковщик, позволяющий вернуть защищенный файл в первородный вид, но после небольшой доработки (спасибо исходным текстам!), приобретает весь необходимый набор защитных методик. Намного лучше доработать Upx, чем использовать любой из существующих навесных протекторов, поскольку всякий клон Upx'а приходится исследоваться индивидуально и хакер не может использовать общие схемы. Естественно, для модернизации упаковщика вам понадобятся антиотладочные приемы, о технике которых мы сейчас и поговорим.

АНТИОТЛАДОЧНЫЕ ПРИЕМЫ

Большинство антиотладочных приемов по своей природе системнозависимы и препятствуют переносу защищенной программы на другие платформы, поэтому пользоваться ими следует с большой осторожностью и осмотрительностью, тщательно тестируя каждую строку кода.

ПАРАЗИТНЫЕ ФАЙЛОВЫЕ ДЕСКРИПТОРЫ

В большинстве (если не во всех) *nix-систем запущенный нормальным образом файл получает в свое распоряжение три дескриптора – 0 (stdin), 1 (stdout), 2 (stderr). GDB и подобные ему отладчики создают дополнительные дескрипторы и не закрывают их. Чтобы обнаружить отладчик, достаточно попытаться закрыть дескриптор №3, и если эта операция завершится успешно, значит, нас отлаживают по полной программе!

Готовый пример реализации может выглядеть, например, так:

```
if (close(3)==-1)
        printf ("all ok\n");
else
        printf ("fuck off,debugger!\n");
```

АРГУМЕНТЫ КОМАНДНОЙ СТРОКИ И ОКРУЖЕНИЕ

Оболочка типа Bash автоматически подставляет имя запускаемого файла в переменную окружения "_". Отладчики же оставляют ее пустой (см. таблицу). Наблюдаются некоторые различия и с нулевым аргументом командой строки: Bash и подавляющее большинство остальных оболочек подставляют сюда текущее имя файла, а GDB – имя файла с полным путем (впрочем, ALD таким путем распознать не удается).

	argv[0]	getenv("_")
shell	./file_name	./file_name
strace	./file_name	/usr/bin/strace
ltrace	./file_name	/usr/bin/ltrace
fenris	./file_name	/usr/bin/fenris
gdb	/home/usr/file_name	(NULL)
acl	./file_name	(NULL)

Распознавание отладчика по параметрам командной строки и переменным окружения

ДЕРЕВО ПРОЦЕССОВ

В Linux при нормальном исполнении программы идентификатор родительского процесса (Ppid) всегда равен и идентификатору сессии (Sid), а при запуске под отладчиком Ppid и Sid различны (см. таблицу). Однако в других операционных системах (например, во FreeBSD) это не так, и Sid отличается от Ppid даже вне отладчика. Как следствие, программа, защищенная по этой методике, дает течь, отказывается выполняться даже у честных пользователей.

Личное наблюдение: при нормальном исполнении программы под FreeBSD идентификатор текущего процесса существенно отличается от идентификатора родительского, а при запуске из-под отладчика идентификатор родительского процесса оказывается меньше ровно на единицу. Таким образом, законченный пример реализации может выглядеть так:

```
main ()
{
   if ( (getppid() != getsid(0))
        && ((getppid() + 1) != getppid())
           printf("get out, debugger!\n");
   else
           printf("all ok!\n");
}
```

СИГНАЛЫ, ДАМПЫ И ИСКЛЮЧЕНИЯ

Следующий прием основан на том факте, что большинство отладчиков жестко держат SIGTRAP-сигналы (trace/breakpoint trap) и не позволяют отлаживаемой программе устанавливать свои собственные обработчики. Как можно использовать это для защиты? Устанавливаем обработчик исключительной ситуации посредством вызова Signal (SIGTRAP, handler) и спустя некоторое время выполняем инструкцию INT 03. При нормальном развитии событий управление получает Handler, а при прогоне программы под GDB происходит аварийный останов с возвращением в отладчик. При возобновлении выполнения программа продолжает исполняться с прерванного места, при этом Handler так и не получает управления. Имеет смысл повесить на него расшифровщик или любую другую "отпирающую" процедуру. »

```
# cat anti-gdb-1.c
#include <stdio.h>

main()
{
        printf("anti-gdb-1 demo by A^C^E\n");
        if (close(3)==-1)
                printf("gdb not detected, all ok\n");
        else
                printf("gdb detected, fuck off\n");

}
#
# /bin/_anti-gdb-1
anti-gdb-1 demo by A^C^E
gdb not detected, all ok
#
# gdb -q /bin/_anti-gdb-1
(no debugging symbols found)...(gdb) run
Starting program: /bin/_anti-gdb-1
anti-gdb-1 demo by A^C^E
gdb detected, fuck off
(no debugging symbols found)...(no debugging symbols found)...
Program exited with code 027.
(gdb)
```

Капкан для debugger'а

	shell	gdb	strace	ltrace	fenris
getsid	0x1968	0x1968	0x1968	0x1968	0x1968
getppid	0x1968	0x3a6f	0x3a71	0x3a73	0x3a75
getpgid	0x3a6e	0x3a70	0x3a71	0x3a73	0x3a75
getpgrp	0x3a6e	0x3a70	0x3a71	0x3a73	0x3a75

Вариации идентификаторов в Linux

Это очень мощный антиотладочный прием, единственный недостаток которого заключается в привязанности к конкретной аппаратной платформе – в данном случае к платформе Intel.

Конкретный пример реализации выглядит так:

```
#include <signal.h>

void handler(int n) { /* обработчик исключения */ }

main()
{
    // устанавливаем обработчик на INT 03
    signal(SIGTRAP, handler);

    //...

    // вызываем INT 03, передавая управление handler'у
    // или отладчику (если он есть)
    _asm_("int3");

    // зашифрованная часть программы,
    // расшифровываемая handler'ом
    printf("hello, world!\n")
}
```

Программные точки останова (машинная команда INT 03h с опкодом CCh) распознаются обычным подсчетом контрольной суммы собственного кода программы.

РАСПОЗНАВАНИЕ ПРОГРАММНЫХ ТОЧЕК ОСТАНОВА

Программные точки останова (машинная команда INT 03h с опкодом CCh) распознаются обычным подсчетом контрольной суммы собственного кода программы. Поскольку порядок размещения функций в памяти в общем случае совпадает с порядком их объявления в исходном тесте, адрес конца функции равен указателю на начало следующей функции.

Контроль целостности своего кода как средство обнаружения программных точек останова

```
foo() {/* контролируемая функция 1 */ }
bar() {/* контролируемая функция 1 */ }
main()
{
    int a; unsigned char *p; a = 0;
    for (p = (unsigned char*)foo; p < (unsigned
char*)main; p++)
        a += *p;

    if (a != _MY_CRC)
        printf ("get out, debugger!\n");
    else
        printf ("all ok\n");
}
```

МЫ ТРАССИРУЕМ, НАС ТРАССИРУЮТ

Функцию Ptrace нельзя вызывать дважды – попытка трассировки уже трассируемого процесса порождает ошибку. Это не ограничение библиотеки Ptrace – это ограничение большинства процессорных архитектур (хотя на x86-процессорах и можно развернуться). Отсюда идея – делаем Fork расщепляя процесс на два и трассируем самого себя. Родителю достается PT_ATTACH (он же PTTRACE_ATTACH), а потомку – PT_TRACE_ME (он же PTTRACE_TRACE_ME). Чтобы хакер не прибил Ptrace, в ходе трассировки рекомендуется сделать что-нибудь полезное (например, динамически расшифровать код), и тогда отладка такой программы будет возможна лишь на эмуляторе.

Простейший пример реализации может выглядеть, например, так:

самотрассирующаяся программа

```
int main()
{
    pid_t child; int status;
    switch((child = fork())) {
    case 0: // потомок
            ptrace(PTRACE_TRACEME);
            // секретная часть
            exit(1);
    case -1: // ошибка
            perror("fork"); exit(1);
            default: // родитель
            if (ptrace(PTRACE_ATTACH, child)) {
                kill(child, SIGKILL); exit(2);
            }
            while (waitpid(child, &status, 0) != -1)
                ptrace(PTRACE_CONT, child, 0, 0);
            exit(0);
    }
    return 0;
}
```

ПРЯМОЙ ПОИСК ОТЛАДЧИКА В ПАМЯТИ

Любой отладчик прикладного уровня может быть обнаружен тривиальным просмотром содержимого /proc. Хороший результат дает поиск по сигнатурам – текстовым строкам копирайтов конкретных отладчиков. Чтобы быть уверенным, что отлаживают именно нас, а не кого-то еще, можно сравнить идентификатор процесса отладчика (он совпадает с именем соответствующей директории в /proc) с идентификатором материнского процесса (его можно получить с помощью Getppid), однако если отладчик сделает Attach на активный процесс, это не сработает.

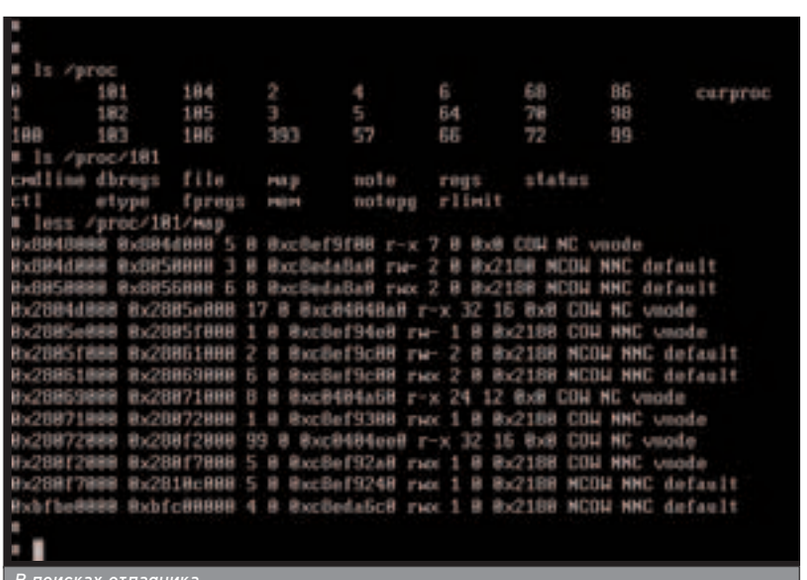
Лучше не заметить отладчик, чем реагировать на отладку посторонних процессов.

ИЗМЕРЕНИЕ ВРЕМЕНИ ВЫПОЛНЕНИЯ

Отладчики прикладного уровня не "замораживают" часы в процессе трассировки, и поэтому измерение отрезка времени между двумя соседними участками программы позволяет обнаружить как отладку, так и шпионаж за системными функциями посредством truss\ktrace.

ЗАКЛЮЧЕНИЕ

Всякая защита лишь отодвигает взлом, но отнюдь не исключает его возможность. Все-таки начиная с некоторого уровня сложности взломов это занятие становится нерентабельным, и единственным стимулом хакерского труда остается спортивный интерес, вызванный природным любопытством и желанием покопаться в интересной программе. Не стремись к элегантности! Используй тошнотворный стиль кодирования, вызывающий у хакера отвращение хуже, чем от горькой редьки. Тогда шансы на выживание у твоей программы значительно возрастут, и долгое время она будет оставаться невзломанной.



В поисках отладчика

БАЗЫ ДАННЫХ

Читай в следующем номере Спеца:

- Теория баз данных
- Моделирование
- Основы работы
- Оптимизация БД и повышение производительности
- ODBC: практика
- Базы знаний
- Генерация отчетов
- Средства разработки
- Базы данных + XML
- Безопасность БД
- Резервное копирование и восстановление
- Уязвимости

А также:
- СУБД MySQL, MS SQL Server 2005, Oracle и еще сотня причин систематизировать свои данные!

Весь софт на CD!

АНОНС

СКОРО В СПЕЦЕ:

- **Взлом и защита программ**

Методы взлома программ. Дизассемблирование, отладка, dumping. Реализация и снятие защиты. Шифрование и сжатие, упаковка. Восстановление таблицы импорта. Защита. Вирусные технологии для защиты от cracking'а. Низкоуровневая и аппаратная защита.

- **Цифровое видео**

Запись, просмотр, монтаж, съемка, раскрутка. Обзор систем, принципы работы, компрессоры, кодеры, декодеры, алгоритмы сжатия, реальные программы, их настройка, спецприемы и крутые эффекты. Тесты производительности, грамотный захват.

- **Коммерческий кодинг**

Способы заработка, связанные с ПО. Как заработать на open source-продуктах. Какие средства и для каких целей используют профессиональные разработчики. Аутсорсинг, взгляд внутрь команд разработчиков. "Шароварение". Юридичесчекие вопросы с учетом отечественной действительности.

- **Мобильные устройтва и их безопасность**

Взлом с помощью мобильных устройств. Bluejacking, bluesnarfing и взлом Wi-Fi-сетей. Сниферы Wi-Fi\Bluetooth. Все о wardriving. Мобильные вирусы и трояны. Security-софт под мобильные платформы. Фрикинг, безопасность в телекоммуникациях. Спам.

- **Интернет-деньги**

Обменники валюты, казино и другие web-сервисы, связанные с интернет-валютой. Различные системы: WebMoney, e-gold, GoldMoney, PayPal g.p. Заработок\процессинг: что и как реализовать. Как сделать свою пирамиду\банк, как кидают в e-бизнесе.

Content:

SPECial delivery

Каролик Андрей (andrusha@real.xakep.ru)

*NIX - ЛИТЕРАТУРА

КНИГИ ДЛЯ *NIX ПОД ПРИСМОТРОМ

Мы и сами привыкли все делать методом тыка и по интуиции. И знаешь, получается. Но это не значит, что мы не читаем книг. В них масса наглядных примеров, полезные советы и способы автоматизации. Другими словами, то, до чего методом тыка не дойти. Мы отобрали те книги, которые могут реально пригодиться тебе.

СЕКРЕТЫ ХАКЕРОВ. БЕЗОПАСНОСТЬ LINUX - ГОТОВЫЕ РЕШЕНИЯ

М.: Издательский дом "Вильямс"
2004
Брайан Хатч
704 страницы
Разумная цена: 300 рублей

Практическое руководство для тех, кто решил обеспечить безопасность своей ОС Linux по-настоящему. Очень подробно рассказывается о мерах защиты как от классических атак, так и от новых средств из арсенала хакеров. Удобно то, что все известные типы атак показаны на реальных примерах. В этой книге каждой атаке - по способу защиты (профилактика и выявление вторжения). Девиз книги - "Научись думать, как хакер, чтобы защитить свою Linux-систему". Книга поможет понять методы, которые используют хакеры, их замыслы и реализацию замыслов. Ты узнаешь, как хакер выбирает цель для атаки, как получает нужную информацию, как использует уязвимые места, как получает контроль над чужой системой, как скрывает свое присутствие и т.п. Отдельно рассмотрены основные ошибки в ПО почтовых серверов, шифрование электронной почты и блокирование спама. И ценные советы из личного опыта работы от авторов книги.

РУКОВОДСТВО АДМИНИСТРАТОРА СЕТИ ОС LINUX

Ярославль: БТИ "Еще не поздно!"
2003
Паутов Алексей
346 страниц
Разумная цена: 340 рублей

ОС Linux быстро завоевала популярность своей гибкостью, надежностью, хорошей масштабируемостью, бесплатностью (как бутерброд с черной икрой для русских халявщиков) и свободным доступом ко всем исходным текстам - если разбираешься в программировании, сможешь править систему под свои задачи, оптимизируя и исправляя как душе угодно. Но Сеть настроить здесь - далеко не как в стеклянных, а с долей небольшого геморроя. С помощью данной книги ты настроишь Сеть с нуля, грамотно настроишь почтовый сервер для работы с кириллицей и обезопасишь машину от сетевых атак.

ПОЛНЫЙ СПРАВОЧНИК ПО FREEBSD

М.: Издательский дом "Вильямс"
2004
Родерик Смит
672 страницы
Разумная цена: 300 рублей

Об инсталляции, конфигурировании и обслуживании FreeBSD подробно. Правда, акцент в книге сделан больше на администрирование, а не на банальное использование. Есть и разделы, посвященные использованию имеющихся в ОС графических и офисных приложений. Устранение проблем совместимости, подготовка системы и инсталляция FreeBSD. Конфигурирование сети,

настройка файловых, почтовых и web-серверов. Модификация, перекомпиляция и обновление ядра. Использование оболочек GNOME и KDE, создание собственной графической оболочки. Безопасность системы.

RED HAT LINUX. СЕКРЕТЫ ПРОФЕССИОНАЛА

М.: Издательский дом "Вильямс"
2004
Наба Баркакати
1056 страниц
Разумная цена: 600 рублей

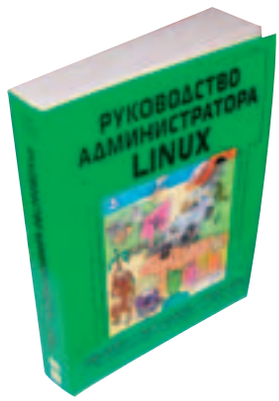
Книга, по сути, похожа на многие другие - установка, настройка и использование Red Hat Linux. Но изобилие реальных примеров и множества секретов, которых нет даже в официальной документации и в интернете, - и она становится ценным приобретением для любой книжной полки. Инсталляция Red Hat Linux на ноутбуки, конфигурирование XFree86 для видеокарт, ручная загрузка драйверов, настройка беспроводных Wi-Fi сетей, обновление и настройка ядра и многое другое. Бонус - два диска с Red Hat Linux 9 Publisher's Edition.

LINUX-СЕРВЕР СВОИМИ РУКАМИ

Спб.: Наука и Техника
2004
Колисниченко Д.Н.
704 страницы
Разумная цена: 200 рублей

По сути, это описание того, как на основе ОС Linux создать сервер нужной конфигурации и функциональности. То есть как организовать Linux-сервер для выполнения конкретных задач (к примеру, сервер для локальной сети, интернет-сервер или сервер удаленного доступа). Есть и уникальные наработки автора: создание Linux-сервера для игрового клуба, запуск Windows-игр под Linux, учет и разделение трафика, система защиты LIDS и т.д. Курс молодого администратора в начале книги поможет тем, кто не силен в сетевых технологиях и сетевых протоколах.

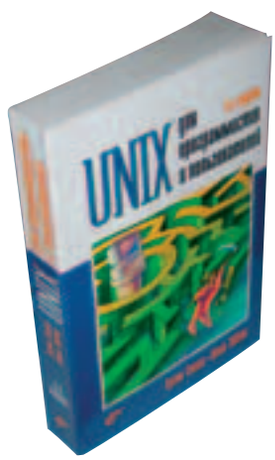
РУКОВОДСТВО АДМИНИСТРАТОРА LINUX



М.: Издательский дом "Вильямс"
2004
Эви Немет
880 страниц
Разумная цена: 230 рублей

Книга посвящена трем основным дистрибутивам: Linux Red Hat 7.2, SuSE 7.3 и Debian 3.0. Выбор на них пал из-за популярности и обширных возможностей, которые присущи и другим Linux-системам. Книга содержит множество конкретных примеров и реальных практических советов по решению нетривиальных задач. Тут и конфигурирование DNS, и сетевое конфигурирование, и настройка электронной почты, и контроль безопасности, и создание системного ядра, и анализ производительности...

UNIX ДЛЯ ПРОГРАММИСТОВ И ПОЛЬЗОВАТЕЛЕЙ



Спб.: БВХ-Петербург
2004
Грэм Гласс
848 страниц
Разумная цена: 260 рублей

Установить UNIX - только полдела. Куда важнее разобраться в командных интерпретаторах, утилитках, библиотечных функциях и взаимодействии процессов. В этой книге ты как раз найдешь все это недостающее. Описаны командные интерпретаторы Bourne shell, Korn shell, C shell и Bourne Again shell. Подробно рассмотрена организация файловой системы, ввод/вывод и взаимодействие процессов. При этом акцент сделан на средства программирования на языке C и на системное программирование. Рассмотрены более 100(!) утилит, включая awk, grep, sed, Perl, vi и emacs. Приведенные примеры и исходные коды сделают чтение простым и приятным.

БЕЗОПАСНОСТЬ LINUX

М.: Издательский дом "Вильямс"
2003
Скотт Манн
624 страницы
Разумная цена: 250 рублей

Если ты в Сети, то безопасность - основная проблема. Даже если твоя машина не представляет особого интереса для взломщиков, надругаться над ней могут просто так. А приобретенные навыки, возможно, пригодятся тебе в будущем, если твоя работа так или иначе будет связана с безопасностью. В книге описаны программы с открытым исходным кодом для защиты Linux: от брандмауэров до аутентификации. Sudo, portmap, xinetd, Bastille, tripwire, ipchains/iptables, crack и многие другие специализированные программы. Известные "ловушки", практические методики и недокументированные приемы. Противостояние троянам, взлому, переполнению буфера и подделке IP-адресов. Обнаружение вторжений с помощью сетевых анализаторов, разработка стратегии защиты, защита электронной почты и многое другое.

■ Любые из описанных книжек, которые тебя заинтересовали, можешь заказать (по разумным ценам) не отрывая пятой точки от дивана или стула в букинистическом интернет-магазине "OS-Книга" (www.osbook.ru), который любезно предоставил нам книжки живьем.

Докучаев Дмитрий aka Forb (forb@real.xakep.ru)

КОМАНДНЫЙ СЛОВАРЬ ЮНИКСОИДА

САМЫЕ ПОЛЕЗНЫЕ КОМАНДЫ

Unix - это в первую очередь сетевая многопользовательская система. В отличие от Windows, эта ОС включает в себя множество команд для самых разных операций. Чтобы без труда ориентироваться хотя бы в базовых запросах, ознакомься с этим командным глоссарием.

ФАЙЛОВЫЕ КОМАНДЫ

Работая в Unix, ты, несомненно, будешь проводить операции над файлами и каталогами. Чтобы не запускать графический менеджер ради только одной операции, тебе потребуются самые распространенные запросы для основных файловых операций.

mkdir <каталог> - создание каталога. Если тебе нужно построить ветку директорий, укажи дополнительный параметр -p.

touch <file> - создание пустого текстового файла. В случае если файл уже существует, его дата и время доступа изменятся на текущие.

rm <file> - удаление файла. Для удаления каталога - ключ -r. Если хочется удалить непустой каталог, прибавляй параметр -f. Но помни, что с помощью "rm -rf" можно случайно уничтожить важную информацию - будь с ней осторожнее.

ls <dir> - отображение списка файлов в заданном каталоге. Более детальную информацию можно получить при использовании добавочных ключей "-alF". Опция "-R" позволяет выполнить рекурсивный просмотр каталогов.

cp file1 file2 (mv file1 file2) - копирование (перемещение) файла. Чтобы скопировать содержимое каталога, укажи параметр -R. Чтобы сохранить права доступа копируемых объектов, укажи флажок -p.

cat <file> - чтение текстового файла. Бинарные файлы читать не рекомендуется: есть большая вероятность сброса настроек терминала :).

cat > <file> и cat >> <file> - запись и дозапись в файл. Символ ">" в данном контексте выступает как перенаправление потока. Действительно, по сути, выполняется запуск cat без параметров, а после нажатия <CTRL>+d (завершение записи) поток с текстом автоматически перенаправится в файл.

head <file> и tail <file> - чтение десяти первых и последних строк файла соответственно. Опции очень полезны при анализе какого-нибудь увесистого лога. Чтобы вывести заданное количество строк, используй опцию "-n число строк".

grep <строка> <файл> - поиск заданной строки в текстовом файле. Чтобы осуществить реверсивный поиск, существует параметр -v. Ключ -i позволяет игнорировать регистр искомого выражения.

ln <file1> <file2> - создание жесткой ссылки в виде file2 на файл file1. Для того чтобы сделать символическую ссылку, нужно добавить ключ -s. Кстати, жесткие ссылки позволительно создавать только на общем дисковом разделе.

pwd - определение текущего каталога.

cd <каталог> - переход в указанный каталог. На самом деле cd - лишь функция shell'а, которая никак не привязана к бинарнику.

tar zcf file.tar.gz [file|folder] - создание архива с файлом или папкой. Извлекается архивчик с помощью замены параметра "c" на "x". Настоящие юниксоиды довольно часто работают с архивами, поэтому запоминай эти опции и присоединяйся к этой славной армии.

find <path> [options] - поиск файла в каталоге <path>, подпадающего под определенный критерий. У этой команды очень много опций, но наиболее простой запуск выглядит так: find /folder -name *filename*.

locate <file> - поиск файла на диске. Эта операция очень быстрая, так как название файла берется из специальной базы, которая ежедневно обновляется.

ПРОЦЕССЫ И ПРАВА

Unix - очень безопасная система, где права пользователя охраняются жестко. Существует ряд команд, которые позволяют смотреть/изменять привилегии и контролировать пользовательские процессы.

id [user] - просмотр собственных прав. На экране увидишь свой UID, групповой идентификатор и все группы, к которым ты принадлежишь. Если добавить в качестве параметра имя системного пользователя, ты сможешь легко посмотреть его привилегии.

chmod <permissions> <file> - изменение прав доступа к файлу. Права могут указываться как в восьмеричной системе, так и символьным путем. Скажем, параметр +x дает право любому на выполнение бинарника. Соответственно, опции +r и +w расставляют привилегии на чтение и запись.

chown <file.group> <file> - изменение владельца файла. Эту команду имеет право выполнить только владелец файла или root.

su [user] - переключиться на другого пользователя. После ввода этого запроса бинарник su потребует ввести пароль администратора (или пользователя, права которого ты хочешь присвоить).

```
[root@sim tmp]# pstree
init-+-CGServer---CGServer---34*[CGServer]
     |-automount
     |-bdflush
     |-crond
     |-drwebd
     |-httpd---10*[httpd]
     |-keventd
     |-khubd
     |-kinoded
     |-kjournald
     |-klogd
     |-kreiserfsd
     |-ksoftirqd_CPU0
     |-kswapd
     |-kupdated
     |-mdrecoveryd
     |-6*[mingetty]
     |-mysqld_wrapper---mysqld---mysqld---2*[mysqld]
     |-named
     |-nmbd---nmbd
     |-portmap
     |-pppd---pptp
     |-pptp
```

Наглядный список процессов

Команда uname -a расскажет много интересного о системе, например, имя операционки, версию ядра или тип процессора.

С помощью запроса set можно смотреть/изменять системные переменные окружения. Например, добавлять новый каталог в переменную PATH.

passwd [user] - смена своего пароля или пароля другого пользователя. Чтобы поменять чужой пароль, необходимо быть администратором системы.

ps - отображение собственных процессов. Чтобы увидеть все системные задания, используй опцию -ax. Если есть желание увидеть имя пользователя, под которым запущен процесс, прибавляй ключик -u.

kill [signal] pid - остановка ненужного процесса. Если после просмотра таблицы ты заметишь процесс, который нужно завершить в принудительном порядке, выполняй команду kill -9 идентификатор. Девятый сигнал невозможно проигнорировать, поэтому pid быстро исчезнет из системной таблицы. Для просмотра доступных сигналов используй команду kill -l.

top - тот же вывод процессов, только в интерактивном виде и в более дружелюбной форме.

pstree - вывод процессов в древовидной форме.

renice pid - изменение приоритета процесса. По умолчанию программа запускается с нулевым приоритетом, однако он может колебаться от -20 (самый высокий) до +20 (самый низкий).

СЕТЕВЫЕ КОМАНДЫ

В консоли имеется огромное количество сетевых клиентов. Важно лишь знать их название и синтаксис.

telnet <host> <port> - подключение на произвольный порт заданного узла. Команда Telnet является универсальным средством сетевого обмена. Никто не запрещает использовать этот бинарник в качестве telnet/smtp/pop3-клиента.

ssh [user@]host - соединение по SSH-протоколу на удаленный узел. Если опустить параметр user@, в качестве логина передается текущее имя пользователя.

ftp <host> [port] - интерактивный ftp-клиент. Является незаменимым средством консольного юниксоида. Для разъяснения параметров напиши Help после запуска клиента.

wget <url> - скачивает файл из глобальной Сети. Wget - самый продвинутый консольный download-менеджер, который устанавливается по умолчанию во многих системах.

lynx <url> или links <url> - консольный web-браузер. Оба они понимают таблицы, фреймы, css, скрипты и многое другое.

ping (traceroute) <host> - посылка пакета icmp-echo (трассировка маршрута) на указанный узел.

host <адрес> - определение IP-адреса символьного хоста.

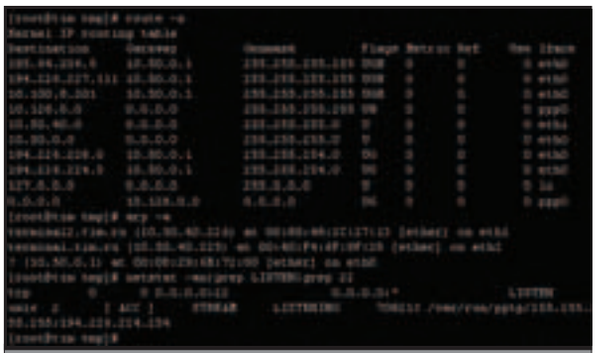
dig <адрес> - то же самое, но в более расширенном формате. Клиент dig имеет множество интересных параметров, является полнофункциональным DNS-клиентом. Смотри Help и просвещайся :).

nmap <host> - твой любимый сканер портов. Кстати, nmap также устанавливается по умолчанию в новых системах. Однако часть опций доступна только администратору сервера.

РАЙ ДЛЯ АДМИНИСТРАТОРА

Наконец, пришло время рассказать о командах, которые чаще всего используются системными администраторами (а ты чем хуже?). Пожалуйста - пользуйся, конечно, если ты хозяин машины.

ifconfig - отображение активных сетевых интерфейсов, параметры которых при большом желании можно изменить. Допустим, чтобы присвоить сетевой карте IP-адрес, достаточно



Жесткий сетевой контроль

написать ipfconfig eth0 ip-address up.

route - просмотр (изменение) таблицы маршрутизации. К примеру, если у тебя два доступа в интернет, нужно грамотно настроить маршрутизацию, чтобы задействовать оба на особо выгодных условиях.

arp - управление ARP-таблицей. Напоминаю: протокол ARP связывает IP- и MAC-адреса.

netstat - отображение сетевой статистики. Без параметров команда покажет активные соединения. Чтобы посмотреть все подключения и открытые порты, используй ключ -a. Для отмены преобразования хостов в IP применяется параметр -n (что значительно ускоряет вывод). Ну и, наконец, чтобы узнать, какой именно процесс привязан к порту, обращайся к опции -p.

iptables - вызов встроенного файрвола. Настройка брандмауэра - тема отдельной статьи, которую ты найдешь в этом журнале :).

useradd <user> [-s shell -d /home/directory -g group] - добавление пользователя с указанными реквизитами.

userdel [-r] <user> - удаление пользователя. Ключик -r позволяет удалить не только учетную запись, но и весь домашний каталог, а также почтовую переписку.

reboot - перезагрузка сервера. Синоним: shutdown -r now.

halt - выключение машины. Синоним: shutdown -h now или poweroff.

СПРАВОЧНАЯ ЛИТЕРАТУРА

Ну и напоследок торжественно передам тебе несколько справочных команд. С помощью справки ты всегда сможешь найти команду, получить список ключей к ней или посмотреть развернутое описание.

man [раздел] <команда> - полное руководство по команде. По умолчанию поиск ведется в первом разделе, однако к одной команде может быть несколько документов. Яркий пример тому - ключевое слово "read" (в первом разделе содержится руководство по ключевому слову "bash", во втором - по функции языка C read).

info <command> [пакет] - более развернутое руководство пользователя. Допустим, запрос info ls coreutls покажет намного больше информации, чем man ls.

apropos <word> - поиск названия руководства по заданному ключевому слову. Например, ты помнишь, что команда содержит подстроку "dir", но не знаешь ее полного имени. Используй запрос apropos dir, и команда быстро найдется.

which <команда> - поиск нужного исполняемого файла в каталогах, объявленных в переменной окружения PATH.

whereis <команда> - эффект тот же, что и для which, только поиск ведется и в каталоге со справочными страницами.

command --help (или command -h) - очень часто параметр -help выводит частичную справку о параметрах команды. Это значит, что совсем не обязательно читать большое руководство, если ты вдруг забыл командную опцию.

Чтобы узнать информацию о залогиненных пользователях, выполни запрос "w". Команда Last поможет подсмотреть данные людей, которые уже вышли из системы.

Если тебе захочется отправить системное оповещение всем пользователям, используй команду wall < файл_с_сообщением или просто echo "сообщение"|wall.

ЧТО ЖЕ ТАКОЕ КОМАНДА?

По сути, команда - это вызов внешнего исполняемого файла. Другими словами, введя в интерпретаторе ключевое слово, ты принуждаешь его найти бинарный файл, соответствующий твоему запросу. Если такой бинарник существует, система пытается его запустить, а иначе выдаст сообщение о том, что команда не опознана. Однако бывают исключения, когда вводимая команда - это исключительная особенность shell'а.

ЗАКАЗ ЖУРНАЛА В РЕДАКЦИИ

Бесплатный телефон по России 8-800-200-3-999 по всем вопросам по подписке

ВЫГОДА

Цена подписки на 20% ниже, чем в розничной продаже!
Разыгрываются призы и подарки для подписчиков
Доставка за счет издателя

ГАРАНТИЯ

Вы гарантированно получите все номера журнала
Единая цена по всей России

СЕРВИС

Заказ удобно оплатить через любое отделение банка.
Заказ осуществляется заказной бандеролью
или с курьером

Стоимость заказа на «Хакер Спец» + CD

115р	за номер (экономия 40 рублей*)
690р	за 6 месяцев (экономия 240 рублей*)
1242р	за 12 месяцев (экономия **620** рублей*)

Стоимость заказа на комплект «Хакер Спец»+CD + «Железо»+CD

189р	комплект на 1 месяц (экономия 85 рублей*)
1071р	комплект на 6 месяцев (экономия 510 рублей*)
2016р	комплект на 12 месяцев (экономия **1250** рублей*)

* экономия от средней розничной цены по Москве

ЗАКАЖИ ЖУРНАЛ В РЕДАКЦИИ И СЭКОНОМЬ ДЕНЬГИ

ПОДПИСНОЙ КУПОН

Прошу оформить подписку:

☐ на журнал Хакер Спец + CD

☐ на комплект Хакер Спец+CD и Железо+CD

на ☐ месяцев

начиная с _____ 200__г.

☐ Доставлять журнал по почте на домашний адрес

☐ Доставлять журнал курьером на адрес офиса (по г. Москве)

Подробнее о курьерской доставке читайте ниже*

(отметьте квадрат выбранного варианта подписки)

Ф.И.О.

дата рожд. ☐☐. ☐☐. ☐☐ г.
(день. месяц. год)

АДРЕС ДОСТАВКИ:

индекс

область/край

город

улица

дом корпус

квартира/офис

телефон ( ) (код)

e-mail

сумма оплаты

Извещение

ИНН 7729410015 ООО «Гейм Лэнд»

ЗАО Международный Московский Банк, г. Москва

р/с № 40702810700010298407

к/с № 30101810300000000545

БИК 044525545 КПП - 772901001

Плательщик

Адрес (с индексом)

Назначение платежа	Сумма
Оплата за «_____» с _____ 200__г. (месяц)	

Ф.И.О.

Подпись плательщика

Кассир

Квитанция

ИНН 7729410015 ООО «Гейм Лэнд»

ЗАО Международный Московский Банк, г. Москва

р/с № 40702810700010298407

к/с № 30101810300000000545

БИК 044525545 КПП - 772901001

Плательщик

Адрес (с индексом)

Назначение платежа	Сумма
Оплата за «_____» с _____ 200__г. (месяц)	

Ф.И.О.

Подпись плательщика

Кассир

КАК ОФОРМИТЬ ЗАКАЗ?

1. Заполнить купон и квитанцию
2. Перечислить стоимость подписки через Сбербанк
3. Обязательно прислать в редакцию копию оплаченной квитанции с четко заполненным купоном любым из перечисленных способов:
   - по электронной почте: subscribe_xs@gameland.ru;
   - по факсу: 924-9694;
   - по адресу: 107031, Москва, Дмитровский переулок, д. 4, строение 2, ООО «Гейм Лэнд», Отдел подписки.

По всем вопросам по подписке можно звонить по бесплатному телефону 8-800-200-3-999.

* Курьерская доставка осуществляется в течении 3х дней после выхода журнала в продажу только по Москве на адрес офиса, для оформления доставки курьером укажите адрес и название фирмы в подписном купоне.

Почтовая подписка

41800 Хакер Спец + CD

Тел.: (095) 974-11-11

16764 Хакер Спец + CD

Тел.: (095) 974-21-31

41800 Хакер Спец + CD

Тел.: (095) 974-11-11

Вы также можете оформить почтовую подписку по каталогам подписных агентств во всех отделениях связи России. Для оформления подписки необходимо знать подписной индекс журнала или найти его в каталоге по названию.

Подписка для юридических лиц

www.interpochta.ru

Москва: ООО "Интер-Почта",
тел.: 500-00-60,
e-mail: inter-post@sovintel.ru

Регионы: ООО "Корпоративная почта",
тел.: 953-92-02,
e-mail: kpp@sovintel.ru

Для получения счета на оплату подписки нужно прислать заявку с названием журнала, периодом подписки, банковскими реквизитами, юридическим и почтовым адресом, телефоном и фамилией ответственного лица за подписку.

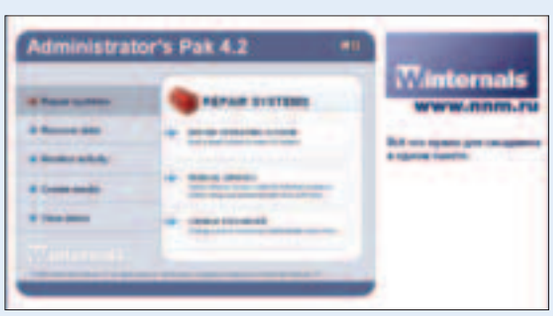
d()c (doc@nnm.ru)

СОФТ ОТ NONAME

Травка зеленет, солнышко блестит, а для тебя пришла пора насладиться свежим софтом от NNM :). Надеюсь, он, как обычно, не оставит тебя разочарованным. Если ты не найдешь на диске пары описанных программ, то уж ссылку на них там ты точно обнаружишь!

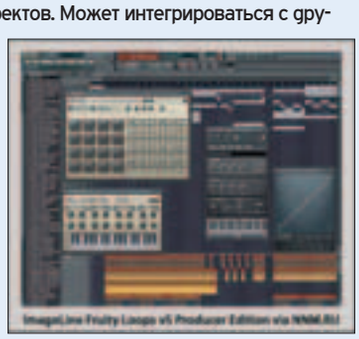
WINTERNALS ADMINISTRATORS PACK V4.2

Идеальный инструмент для системного администратора Windows-платформ. Полная retail-версия, мечта сисадмина :), да и обычным пользователям будет полезно взять на вооружение. Коротко перечислю его выдающиеся способности: восстановление незагружаемых систем, восстановление поврежденной информации, диагностика проблем, связанных с Windows. Administrator's Pak также включает в себя ERD Commander 2003, Disk Commander, NTFSDOS Professional, Remote Recover, Monitoring Tools, TCPView Pro.



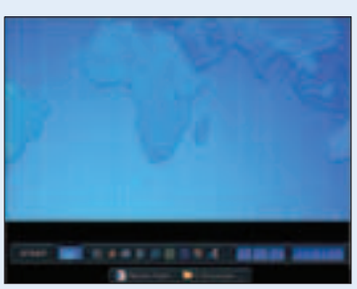
IMAGELINE FRUITY LOOPS V5 PRODUCER EDITION

Свеженький релиз. Установи и используй - все уже подготовлено :). FL Studio - простой в изучении, но очень мощный инструмент для создания музыкальных произведений, поддерживает различные стандарты аудиоплагинов (VSTi, DXi, ReWire), позволяет играть композиции на внешней midi-клавиатуре, содержит кучу инструментов и эффектов. Может интегрироваться с другими средствами - ReBirth, Cubase, Reason...
Хотя он скорее и для "домашних пользователей", но все-таки убедись в том, что он позволяет сделать все, что пожелаешь ты.



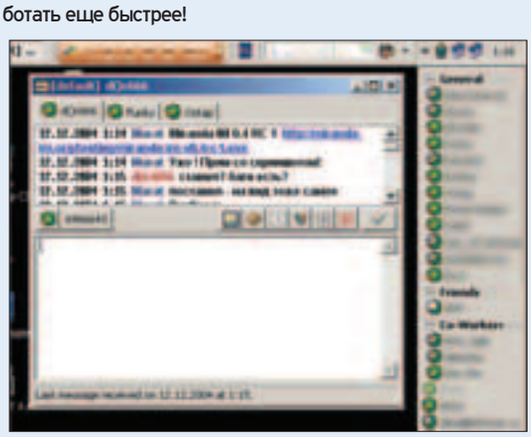
TALISMAN DESKTOP V2.81.2810

Одна из самых известных и "трудно доставаемых" программ для смены внешнего вида Windows. StyleXP и другие аналоги просто нервно курят в сторонке. Талисман не просто меняет "кнопочки" на более округлые или раскрашивает окна в разные цвета, а переворачивает сам принцип расположения всего, что стало для тебя привычным. Довольно много скинов, причем все отличаются друг от друга - каждый подберет себе по душе. Качаем дистрибутив.



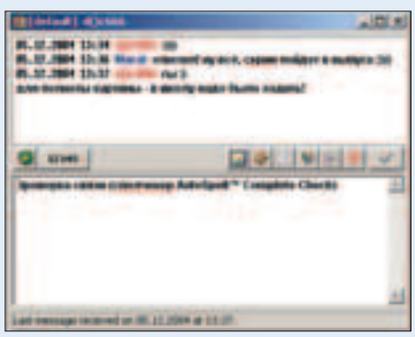
MIRANDA IM 0.4 RC 1

Вышла новая версия самого привлекательного и перспективного (на данный момент) ICQ-клиента - Miranda IM 0.4 RC 1! Качаем (960 Кб)! За что любим, так это за высокую скорость работы и замечательные возможности настройки на свой вкус. На официальном сайте куча плагинов - просто рай для человека, который охотно потратит часок-другой на настройку ради последующего наслаждения полным комфортом.
Я, например, пользуюсь плагином tabSRMM, который позволяет вести все разговоры в одном окне. Клиент поддерживает все системы интернет-пейджинга: ICQ/MSN/AIM/IRC/etc. И самое главное: Miranda стала работать еще быстрее!



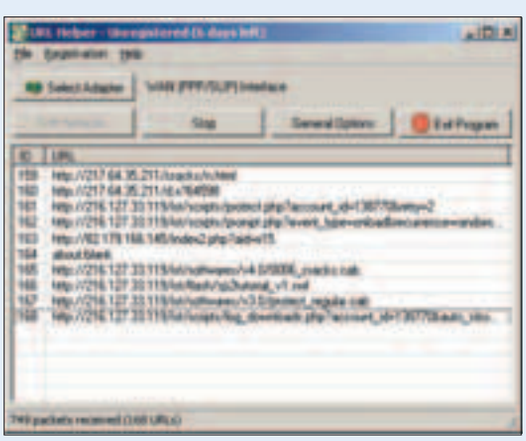
AUTOSPELL COMPLETE CHECK V6.2

Тот счастливчик, который хоть раз работал за Mac'ом, имел шанс заметить его наиполезнейшую особенность - сквозную проверку орфографии (неважно, в какой программе ты набираешь текст). Наконец-то найдено что-то похожее для PC! Вернее, эта утилита появилась давно, но из-за отсутствия поддержки русского языка у нас ее почти не знают. И вот она появилась на просторах нашей необъятной! Некто M.J.Ash (из журнала "Хакер", кстати ;) изловчился и локализовал программку. Встречаем бурными аплодисментами.
Замечательная штука. Подчеркивает ошибки, а в меню по правой кнопке мыши предлагает замену. Работает в Edit'ах любой программы, которую ты используешь. Внимание: для работы необходим установленный MS Office. Инструкция по инсталляции: качаем дистрибутив, инсталлируем. Запускаем AutoSpell Control Panel -> Settings, переходим на вкладку Advanced Settings и кликаем по кнопочке Add. В появившемся окне выбираем язык (Russian (Russia)). В поле Engine Driver Location прописать файл ms97d.dll (C:\Program Files\Autospell60\common files), в поле Engine Location прописать файл mspru32.dll (C:\Program Files\Common Files\Microsoft Shared\Proof), а в Dictionary Location указать файл Msgr_ru.lex, лежащий в той же директории. После этого следует кликнуть OK, вернуться в исходное меню, выбрать русский язык и сделать его языком по умолчанию. Все! Демо-версия программки работает 25 дней, лекарство в розыске.



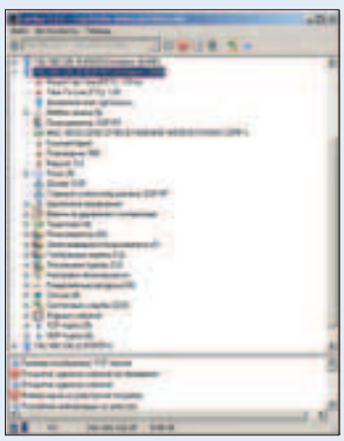
URL MONITOR V1.0

Появилась архиполезная программа - URL Monitor. Часто встречаются сайты, где прямая ссылка для скачивая прячется за сотнями скриптов. Как узнать точный URL всех файлов с сайта, подскажет URL Monitor.
Настройка всего одна: выбираешь сетевой интерфейс, за пакетами которого будем следить, а дальше ходим по сайту. Все URL'ы программа выдирает и вставляет в свой список. Будет полезна всем, кто посещает сайты со взломами, сайты с защищенным медиаконтентом и т.д.



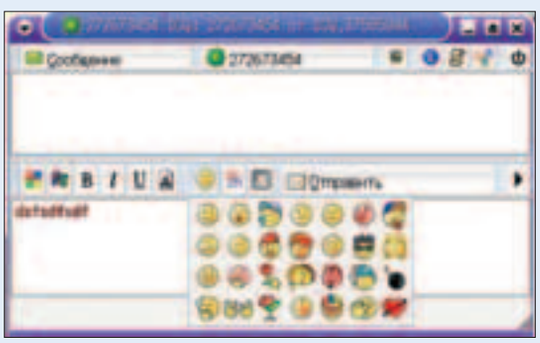
ДЖЕНТЛЬМЕНСКИЙ НАБОР ДЛЯ СИСАДМИНА

...и не только :). Freeware-комплект. На все про все семь метров.
В комплекте:
LanScope 2.9.1 - многопотоковый сканер NetBios- (разделяемых) и FTP-ресурсов. Сканирует заданные диапазоны адресов и определяет доступность ресурсов (чтение, запись). Позволяет искать ресурсы с заданным именем (Music, Video и т.п.). Определяет наличие установленных сервисов (ftp, http) на удаленном хосте.
LanSpy 1.2.1 - LanSpy - это сканер компьютеров в сети, который позволяет получать различную информацию о компьютере.
LanSend 1.3 - LanSend - позволяет отправлять сообщения на компьютер или группу компьютеров в реальном режиме времени. Навороченный net send, одним словом.
LanSafety 1.0 - LanSafety - эта программа поможет тебе установить параметры Windows таким образом, чтобы твоя работа в сети стала более безопасной.
LanShutDown 3.0 - LanShutDown - этот программный пакет позволит тебе выключить питание или перегрузить компьютеры под управлением W2K/XP по сети.
LanLoad 0.9.4.1 - LanLoad - менеджер закачек в локальных сетях. LanLoad предназначен для копирования файлов (папок) в локальных сетях с неустойчивой связью между компьютерами (т.е. в русских локальных сетях - прим. SkyWriter'а).
LanCalculator 1.0 - LanCalculator - это программа, которая позволит тебе без труда рассчитать диапазон адресов в подсети и маски подсети, а также широковещательный адрес, адрес сети, префикс сети и инверсию маски сети, которая используется в списках доступа (ACL) сетевого оборудования Cisco.



SIM 0.9.3

Кто спрашивал про SIM? SIM (Simple Instant Messenger) - это альтернативный ICQ-клиент, который распространяется под лицензией GNU GPL 2 (а значит, и в исходниках, и под все ОС).



Очень удобный QT-интерфейс, стандартный набор функций (прием/передача файлов, SMS, чат и т.д.). Что-то среднее между обычным клиентом ICQ (ненавороченной Мирандой) и навороченной Мирандой :).

Content:

Алексей Малашин, test_lab (test_lab@gameland.ru)

МАЛЕНЬКИЙ ПЕЧАТАЮЩИЙ КОМБАЙН

ТЕСТИРУЕМ ПРИНТЕР SAMSUNG ML-1520P

Samsung выпустил новый принтер для домашнего или малоофисного использования - модель ML-1520P, призванную стать помощником в переводе электронной документации в осязаемый (бумажный) формат. Эта новинка может заинтересовать не только своими рабочими характеристиками, но и ценой, весьма привлекательной для такого класса устройств (всего порядка $170).

Принтер предназначен для работы в SOHO-окружении (Small Office Home Office) и выдерживает нагрузку до 15000 отпечатков в месяц, что прекрасно обеспечит как малый документооборот (в офисе), так и распечатку различных мануалов и книжек (для домашнего использования). Для устройств такого класса немаловажна конечная стоимость одного полученного с принтера листа (для данной модели составляет около двух центов без учета бумаги, а в режиме экономии тонера можно снизить этот показатель почти вдвое). Конечно же, выгодная цена - одно из главных преимуществ, но не стоит сбрасывать со счетов и качество получаемых отпечатков, и длительности/надежность работы устройства. У Samsung ML-1520P и то, и другое на высоте. Несмотря на монохромность и малое разрешение печати, есть возможность получать некую информацию о фотографиях, полученных при помощи цифровой техники: довольно удобно создавать thumbnail-предпросмотры картинок с компьютера для их последующей полноцветной распечатки с помощью фотопринтера. В режиме экономии тонера наблюдается небольшая "бледноватость" материалов, однако это совершенно не мешает чтению текста и различению начертания шрифтов. Особенно порадовало отсутствие "замыливаний" при большом количестве черного цвета на листе, что довольно часто допускают аналогичные модели других производителей. Имеющейся в наличии памяти размером 8 Мб и процессора с частотой 150 МГц должно хватить на одновременное обслуживание простых документов нескольких пользователей, однако при выводе больших картинок возможна некоторая задержка обработки принтером полученной информации. Еще одной приятной особенностью является малая шумность аппарата (и наличие режима экономии энергии), поэтому при постоянной работе устройства даже в непосредственной близости от него дискомфорта не возникнет.

Долговечность и надежность принтера подтверждает фирменная гарантия, предоставляемая производителем. Срок действия гарантии составляет целых три года, правда, из них в течение двух лет гарантировано сервисное обслуживание.

В итоге получаем дружественный принтер, способный стать достойным помощником дома и в небольшом офисе, причем неплохой набор возможностей как по совместимости с аппаратной, так и с программной частью должен облегчить первоначальную установку устройства и работу с ним.

Техническая спецификация:
Модель: Samsung ML-1520P
Скорость печати при 5% заполнении, сек: 14
Максимальное разрешение печати, тчк/дюйм: 600x600
Емкость тонера, листов 5% заполнение: 3000
Рабочий цикл, стр/месяц: 15000
Время выхода первой страницы, сек: 12
Процессор, МГц: Samsung 150
Базовая память, Мб: 8
Набор шрифтов: Windows
Язык управления печатью: SPL (Samsung Printer Language)
Поддерживаемые ОС: Windows 9x/Me/2000/XP, Linux
Интерфейсы подключения: IEEE 1284 (LPT), USB 1.1
Физические размеры, мм: 348x355x193
Вес, кг: 7

Алексей Короткин, test_lab (test_lab@gameland.ru)

СТАРАЯ ПТАШКА В НОВОМ ОПЕРЕНИИ

THRUSTMASTER FORCE FEEDBACK JOYSTICK

Под конец года (24 ноября 2004 года) компания Thrustmaster представила в России новый игровой манипулятор Force Feedback Joystick. По исполнению и внешнему виду эта модель является продолжением линейки Top Gun. Дизайн по традиции техногенный и даже, можно сказать, футуристический. Отличительной особенностью этой модели является наличие обратной связи (Force Feedback), реализованной на основе технологии TouchSense фирмы Immersion. До этого мы тестировали только обратную связь Top Gun Afterburner Force Feedback, который выполнен в виде HOTAS (джойстик плюс рукоятка "сектора газа" с кнопками под пальцами). У Top Gun Afterburner Force Feedback блок "сектора газа" можно отсоединить, и тогда его корпус становится идентичен Force Feedback Joystick, но при этом становятся недоступны "сектор газа" и throttle, отвечающий за ось Z (в авиасимуляторах соответствует "рысканью"). В новой модели для этого реализовано вращение ручки вправо-влево, а "сектор газа" выполнен в виде рычажка на основании манипулятора. Количество функциональных клавиш увеличилось до восьми (добавилась дополнительная кнопка под левую руку), однако теперь элементы управления на основании сделаны вровень с поверхностью корпуса (раньше кнопки были выпуклые), что затрудняет их поиск "вслепую". Для хардкорных авиасимуляторщиков этого маловато, и все равно придется нырять к клавиатуре, но для "среднего" геймера такого количества кнопок вполне достаточно (и даже с избытком). Рукоятка осталась старой доброй: пальцы удобно располагаются в пазах, а кисть упирается в подставку и не устает, шероховатая пластмасса, из которой сделана рукоятка, не позволяет руке скользить и потеть. Что же касается элементов управления, то третья кнопка (слева под указательным пальцем) хоть и стала более выпуклой, но слабовата, что приводит к случайным и двойным нажатиям, а следовательно, к случайным выстрелам и нерациональности расхода боеприпасов. Также смущает наличие только одного 8-позиционного переключателя (используется для переключения обзора) – в симуляторах современных самолетов лучше запастись двумя.

Конструкция манипулятора довольно прочная: основание рукоятки укреплено металлическим кольцом и снабжено резиновым клапаном для защиты от пыли. Размеры для девайса с Force Feedback вполне компактные, однако нас расстроил большой и тяжелый внешний блок питания, необходимый для снабжения энергией сервоприводов. В процессе работы устройство заметно нагревается. Основание джойстика снабжено резиновыми ножками, которые не дают ему скользить по столу. Даже в самые "жаркие" моменты игры манипулятор не будет отрываться от поверхности стола.

Интерфейс подключения - USB. Устройство без проблем устанавливается в систему и не требует никакой настройки. В драйвере можно только протестировать работоспособность элементов управления и поиграть с эффектами обратной связи. Впрочем, необходимые настройки можно произвести и средствами авиасимулятора.

Мы протестировали Thrustmaster Force Feedback Joystick в "Ил-2 Штурмовик". Дефолтные настройки элементов управления (за исключением кнопок на основании) позволяют сразу приступить к выполнению игровых заданий. При входе в миссию джойстик автоматически центрируется, но не всегда точно – приходится поправлять. При настройках "по умолчанию" в центральном положении имеется некая "слепая" зона, за пределами которой начинает действовать обратная связь, из-за чего происходит резкий переход и управление не отличается плавностью (соответственно, в игре машину "бросает" в соответствующую сторону). Все-таки, помучив кривые отклика и немного приноровившись, можно добиться плавного управления.

Feedback реализован на высоте: сервоприводы достаточно мощные, а эффекты четко соответствуют игровому событию (попадания в самолет из крупнокалиберных пулеметов и турбулентность ощущались очень реалистично).

В целом, новая модель производит хорошие впечатления. Ее можно рекомендовать как универсальное решение для геймеров. Манипулятор отлично подойдет как для "реалистичных" авиасимуляторов, так и для фантастических и космических симов.

Интерфейс: USB
Внешнее питание: есть
Force Feedback: есть
Управление: 4 кнопки на рукоятке, 4 кнопки на основании, 8-позиционный переключатель (Hat-свич), 1 throttle в основании рукоятки, ось Z – вращение рукоятки.
Размеры: 24x25x19.

NeO1eX (NeO1eX@rambler.ru), www.alexhak.narod.ru

ПАЯЛЬНИК

МАГНИТНЫЙ ДЖОКЕР

А не сыграть ли нам в картишки? Безусловно, материальных воплощений карт великое множество, а потому играть мы будем с чем-то более экзотическим - с картами магнитными.

INTRO

Куда катится мир? Еще десять лет назад магнитные карты в России если кто и видел, то только на отсканированной на ручном сканере фотографии. Ручные сканеры раздавила аршинная лапа тотальной компьютеризации, выплеснув на рынок планшетные сканеры, а вот магнитные карты не только не потеряли своей актуальности, но и расплодились со скоростью, которой бы позавидовал любой кролик-стахановец. Они везде: и как средство оплаты (банковские платежные и кредитные карты, карты для оплаты проезда в метрополитене), и как средство контроля (электронные пропуска, проездные билеты - в России не встречал, но за бугром довольно распространенное явление). Естественно, такую интервенцию не обделил вниманием российский андеграунд, который тоже захотел внести свою лепту в НТР. И так как у этого андеграунда своя точка зрения на любой предмет, то и лепта получилась соответственная. Мало-помалу из этих under и ground слепились два отдельных направления, специализирующихся на магнитных картах. Для одних был важен сам процесс, для других - результат. Одни искали ошибки в работе, другие применяли их как средство обогащения самих себя. Однако остальному "не-андер" это безусловно условное деление было по условному барабану. Недолго думая, эти неандертальцы по свойственной только им простоте душевной обозвали эти два принципиально разные направления одним коротким словом - carder. Что было дальше, ты знаешь... От себя хочу добавить только одно: есть кардеры-жестянщики, а есть мошенники, использующие технические средства для удовлетворения своих животных потребностей. Определись с тем, кто ты.

УСТРОЙСТВО МАГНИТНЫХ КАРТ

Пообтершись на западе, различные магнитные карты хлынули к нам, на восток. В этой терке участвовало несколько форматов, но повсеместное применение нашел почему-то только один - ID-1, который соответствует стандарту ISO 7810 и описывает габаритные размеры карт. Стандарты ISO 7811-4 и ISO 7811-5 определяют расположение магнитных дорожек, стандарт ISO 7813 определяет спецификации для банковских транзакций, а стандарт ISO 7812 описывает механизмы контроля. Конечно, это далеко не все "исошники", которым должна соответствовать легальная магнитная карта, однако это те из них, которых стараются придерживаться поставщики услуг у нас в России.

Структура магнитной карты показана на рис. 1: три магнитные полосы разделены защитными промежутками. Магнитная полоса банковских карт изготовляется из мелкодисперстного (что-то вроде пудры) ферромагнетика, который вкрапляется в пластик в процессе производства карты. Конечно же, это не единственная технология нанесения магнитных дорожек. Все чаще и чаще встречаются магнитные карты (вполне легального происхождения), на которые дорожки нанесены уже на последнем этапе изготовления - их просто рисуют лаком, в основу которого опять же входит мелкодисперстный ферромагнетик. Ну а заядлые картежники в качестве всех трех магнитных полос используют обычную ленту с пожеванной видеокассеты (на этом и палятся). В области защитного промежутка может быть все что угодно, в том числе тот же ферромагнетик. Промежуток нужен только для учета разброса параметров считывающей головки, чтобы избежать случайного считывания не с того трека. Раз пошла речь о дорожках, давай рассмотрим каждую по отдельности. Как видно по **рис. 1**, каждая дорожка имеет свое условное

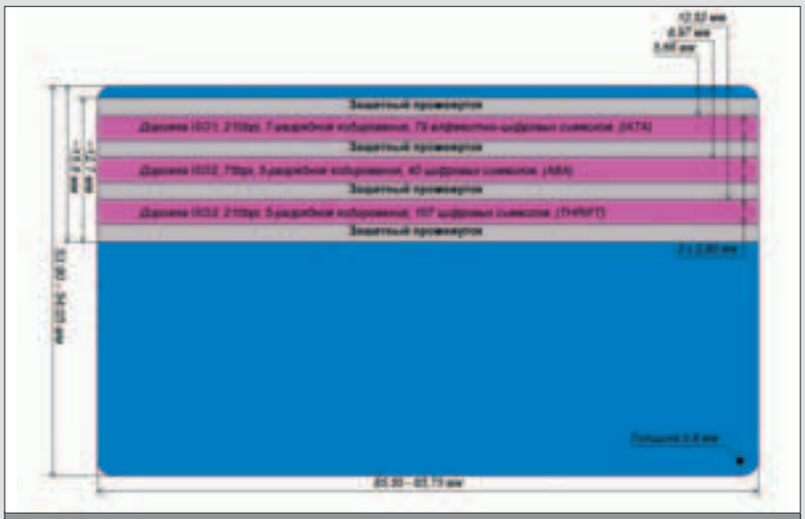


Рис. 1. Расположение магнитных дорожек на карте

Как видно по **рис. 1**, каждая дорожка имеет свое условное обозначение.

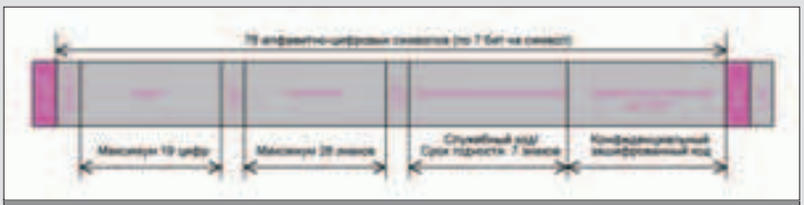


Рис. 2. IATA (iso1)

обозначение (iso1, iso2, iso3). Из всех условных обозначений эти - самые условные, потому что не содержат в себе никакой информации. Видимо, поэтому за ними закрепились более характеризующие названия.

IATA (ISO1)

Изначально на этой дорожке размещали данные, необходимые для резервирования авиабилетов. Название является аббревиатурой забугорной унии авиаперевозчиков, в которую "Аэрофлот", видимо, не входит. Плотность записи составляет 210 bpi (бит на дюйм). Структура содержимого дорожки показана на **рис. 2**. Если сравнить данные о количестве знаков на рис. 1 и рис. 2, то можно найти небольшую разницу в цифрах. Ошибки тут никакой нет, просто реально используются не все знаки. Исходя из простой логики, делаем вывод (который легко подтверждается на практике), что данные располагаются ближе к центру дорожки и по краям заполнены лог.0. Начало информативного содержимого определяет специальный стартовый знак. За ним следует поле "Формат", которое часто определяет назначение карты. Поле "Номер" определяет индивидуальный порядковый номер карты. Поля SEP (separator) служат разделителями и содержат по одному знаку чистых лог.0. В поле "Название" помещена информация об изготовителе. В следующие два поля помещается специфическая информация (о сроке годности, коде доступа и т.п.). Содержимое этих полей напрямую зависит от поля "Формат". Поле "Конец", как и "Старт", состоит из одного спецзнака. Апофеоз всего этого - содержащий один знак блок LRC, обозначающий контрольную сумму всех знаков, вычисленных путем последовательного применения лог. операции XOR (исключающее "ИЛИ").

ABA (ISO2)

Несмотря на то, что дорожка IATA (**рис. 3**) единственная, в которой используется семибитное кодирование, дорожка ABA чаще других используется в качестве носителя банковской информации. Назначение полей ничем не отличается от дорожки iso1. Довольно широко распространены карты, на которых дорожка iso2 является открытым ключом для поля "Дискретизированные данные" дорожки iso1 (в принципе, ничто не исключает обратного). Отличия структуры очевидны - пятибитное кодирование и количество знаков на дорожке.

THRIFT (ISO3)

Эта дорожка (**рис. 4**) названа в честь конторы, которая рекомендует использовать ее. Так как контора маленькая, а ленивых банкиров много, то эта дорожка в банковской сфере практически не используется. А если и используется, то единственным обязательным параметром является относительное расположение блоков на дорожке и пятибитное кодирование данных. Эта дорожка - первый кандидат на занесение в Красную книгу, а кандидатом она является только по одной причине: на ней располагается информация о количестве оставшихся поездок (бесплатных =)) в метро. Тема "бесплатных" поездок подробно рассматривалась в одном из прошлых номеров журнала "Хакер". Таблицы истинности для пяти- и семиразрядного кода ты найдешь на диске.

Карты всякие нужны, карты всякие важны. Конечно, обобщенная информация по магнитным дорожкам нужна и востребована, однако чаще бывают интересны конкретные данные, распространяющиеся на конкретные типы карт. О каждом типе отдельный разговор.

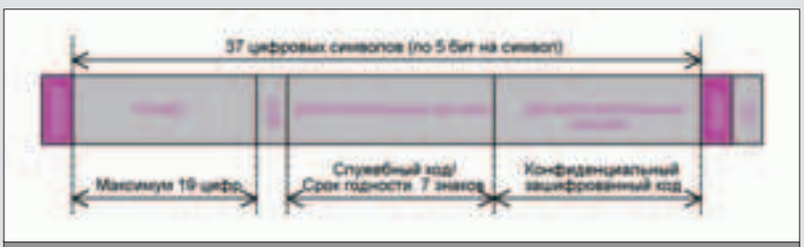


Рис. 3. ABA (iso2).

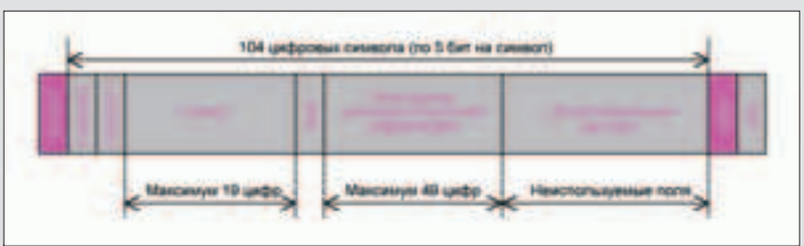


Рис. 4. THRIFT (iso3)

ВСЯ ФИШКА В БАНКЕ

С тех пор как я стал клиентом одного банка, меня стали мучить запор и диарея. Эти безусловно противоположные неудобства были вызваны интересом к внутреннему содержимому любезно предоставленной мне магнитной карты. Вернее, одно неудобство возникло из-за наложенного на карту (мною же) вето, а другое - из-за самой карты. Дело в том, что моя карта соответствует стандарту Visa Maestro, тогда как накопленный анграундом всего мира опыт, как правило, применим лишь к American Express/MasterCard/Discovery. Из представителей торговой марки Visa в этом опыте упоминаются только Visa Electron/Visa Gold и подобные им латинские иероглифы (чтоб не напроситься на гневные реплики в свой адрес, замечу: все описанное только что справедливо лишь для круга моих знакомых). Мне пришлось собирать и анализировать противоречивую информацию и проводить собственные исследования с чистого листа.

Итак, что же конкретно удалось узнать про карты Visa Maestro? Visa Maestro, как и Cirrus, относятся к дебетовым картам и имеют один формат хранения данных - ECCF (European Common Clearing Format), которым, конечно же, пользуется не только Visa и не только Cirrus. Из-за отсутствия интереса к другим картам все изложенное ниже справедливо в первую очередь только для Visa и только для Cirrus, потому что получена в боях с картой, владельцем которой являюсь я. Из-за простоты формата (а может, по другим, непонятным для меня причинам) пользователь такого типа карт мало где может их использовать. Дело даже не в распространенности формата: он широко распространен как в России, так и за рубежом. Формат изначально разрабатывался под нужды малоимущего европейца с его малоимущими незамудренными запросами. А так как малоимущему европейцу излишние расходы (наподобие доступа к платным порносайтам =)) ни к чему, то и сфера применения карты узкая. Например, с ней нельзя оплачивать покупки через web. По-моему, с этой карточкой можно только покупать вещи в магазинах типа "Секонд Хенд". Крупные магазины такими картами брезгуют, предпочитая "Голду" или VISA ELECTRON. Возможно, я утрирую, но опыт есть опыт, и его никуда не денешь.

»

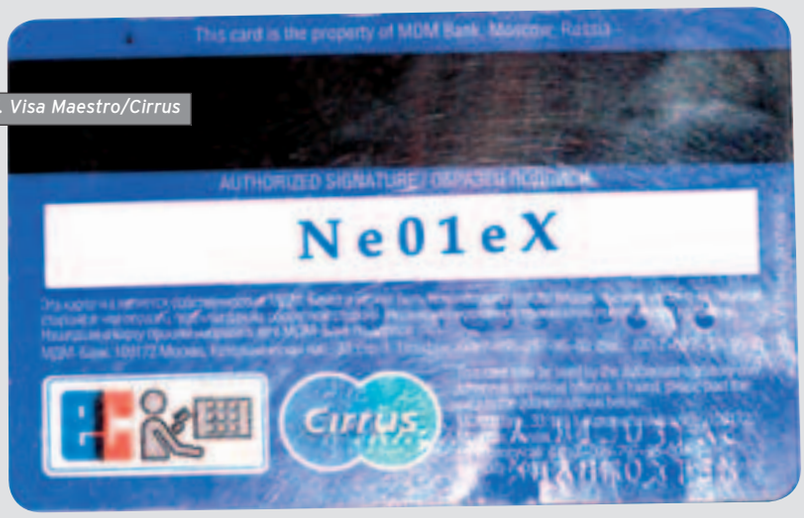


Рис. 5. Visa Maestro/Cirrus

Про внутреннее содержимое карты тоже удалось узнать кое-что. На дорожке iso1 расположена следующая информация:

start B 6764XXXXXXXXXXX sep 0000VISA MAESTRO sep 000MMYY personal_data end lrc, rge

start, sep, end, lrc - стандартные поля;

B - символ поля FORMAT, указывающий на тип карты (справедливо для всех дебетных карт);

6764XXXXXXXXXXX - индивидуальный номер карты;

0000VISA MAESTRO - название карты (первые четыре символа пробелы);

000MMYY - дата окончания срока действия (первые три символа пробелы, дата в формате месяц/год);

personal_data - персональные данные о владельце (ФИ, без отчества), зашифрованные алгоритмом DES. Ключом является пин-код =).

Что это значит? Для использования карты не обязательно знать ее пин-код, достаточно получить в свое распоряжение ФИО владельца и путем дешифрации "от обратного" получить этот самый пин-код. Можно вообще ничего не знать, а воспользоваться каким-нибудь брутфорсером. М-да. Поля дорожки iso2 в основном дублируют поля iso1. Исключением является поле дискретизированных данных, которое содержит 20 цифровых знаков (по-моему, индивидуальных для каждой карты). Дорожка iso3 в дебетных картах Visa Maestro/Cirrus не используется. На **рис. 5** показано фото карты Visa Maestro, которым и заканчивается разбор содержимого банковских дебетных карт.

УТРОМ СТУЛЬЯ - ВЕЧЕРОМ ДЕНЬГИ

Безусловно, рассказом о дебетных картах я не ограничусь, поскольку есть еще карты кредитные, а о них грех не написать. Конечно же, ассортимент кредитных карт довольно велик, но мы рассмотрим лишь карты банка "Русский Стандарт", тем более я на них давно зуб точу (несмотря на то, что являюсь почетным клиентом: обдирают, как козлы капусту). Но тем не менее, эти данные неполные и требуют уточнения.

Внутреннее содержимое карты немного отличается от карты из предыдущего примера. Например, на дорожке iso1 расположены данные в следующем формате:

start B 6219XXXXXXXXXXX sep RUSSKIY STANDART sep MMYY personal_data end lrc.

Алгоритм шифрования дискретизированных данных узнать не удалось, однако не удивлюсь, если ключом окажется пин-код. Поля дорожки iso2 в основном дублируют поля iso1. Содержимое поля дискретизированных данных узнать не удалось, так как пин-код мне не известен (не пошел получать письмо на почту, о чем сейчас жалею). Могу только предположить, что там находится номер лицевого счета. Окольными путями удалось узнать, что на дорожке iso3 расположен ИНН клиента (поле "Номер", 12 символов) и выдаваемая в кредит сумма (поле прикладных данных), причем поле прикладных данных перезаписываемое. Поле дополнительных данных отсутствует. Фото карточки дано на **рис. 6.** Если говорить абстрактно, на всех кредитных картах должна располагаться информация о сфере ее применения. Обычно это трехзначный числовой код, расположенный в поле дискретизированных данных.

ПОСТОРОННИМ В

В наши времена, когда космические корабли бороздят просторы Большого театра, тотальная компьютеризация стучится форточками в каждую дверь. Не за горами тот день, когда в каждой квартире будет стоять электронный замок, а каждый вор будет кардером. Ну а пока этот день не наступил своей железной пятой на обывателя, крупные предприятия стараются его приблизить, вводя в обиход электронные пропуска. Вот и я в определенный момент своей жизни стал его счастливым обладателем и, естественно, не замедлил заглянуть в его магнитное чрево. При просмотре вооруженным (лупой) глазом выяснилась технология изготовления пропуска. Жесткой основой карты являются две полипропиленовые пластинки шириной около 0,25 мм. Внутри располагается обыкновенная (я бы даже сказал, дешевая), сложенная вдвое бумажка с приклеенной фотографией владельца и напечатанными тонером опознавательными знаками (вполне обычными - цех, табельный номер, ФИО). В качестве информационного носителя используется магнитная лента (вполне возможно, от бытовой видеокассеты), расположенная внутри бумаги. Способ ее фиксации выяснять не пробовал, однако не удивлюсь тому, что она тоже просто приклеена. Для обеспечения правильности ввода информации в картоприемник с одной

> Если говорить абстрактно, на всех кредитных картах должна располагаться информация о сфере ее применения.

Рис. 6. Кредитная карта

стороны карты нарисована (тем же самым тонером) черная полоса, имитирующая магнитную ленту. Ширина магнитной ленты немного больше, чем нужно (см. рис. 1), так как готовый пакет сваривается вручную. Разметка карты производится после ее окончательного изготовления. Разметка соответствует рис. 1, однако используются не все поля. В общем виде содержимое дорожки iso1 выглядит в так:

START C ПППТТТТТТ SEP ППDATALOGIC SEP YYYMMDD ФАМИЛИЯ_ИМЯ_ОТЧЕСТВО (до_47_символов) END LRC, где

START, SEP, END, LRC - стандартные кодовые знаки;

C - сфера применения карты, в данном случае - пропуск (не знаю, почему именно "C");

TTTTTT - табельный номер;

DATALOGIC - контора-изготовитель оборудования;

YYYMMDD - дата окончания срока действия пропуска. Используется только для сотрудников, работающих по временному договору (у остальных это поле забито пробелами);

ФИО - ФИО - если полностью не входит, жертвуют отчеством;

П - пробел (символ 0h).

Заметь, что поле дискретизированных данных заполнено этими данными в незашифрованном виде. Дорожка iso2, насколько я понял, не используется: на ней отсутствуют даже стандартные спецзнаки. Дорожка iso3 содержит следующее:

START C F ПППТТТТТТ END LRC, где

START, C, ПППТТТТТТ END, LRC - то же, что и выше;

F - код допуска (на временных пропусках - A);

Код допуска обеспечивает ряд сервисных удобств:

A - ограничение всего, что возможно, проход только через центральную проходную;

F - дает право расплачиваться в кредит в столовой, в пивном баре (за территорией), в магазине, проход через все КПП и т.п. По умолчанию на всех бессрочных пропусках стоит F, однако для того чтобы иметь доступ к этим удобствам, владельцу пропуска нужно заполнить соответствующие бланки с заявлениями.

Этот формат пропуска распространяется на все группы предприятий СУАЛ, включенные в ее состав до 1998 года. Про предприятия, вошедшие в этот алюминиевый консорциум позже, не знаю ничего. Внешний вид электронного пропуска показан на **рис. 7**.

Ты останешься жестянщиком, который знает, что такое честь и честно заработанные деньги.

Рис. 7. Электронный пропуск

Итого

Приведенной выше информации достаточно не только для того, чтобы пользоваться с максимальной эффективностью определенным типом карт, но и для того, чтобы изготовить дубликат. Однако я не рекомендую этого делать, чтобы не попасть под статью о фальсификации документов. Да-да, магнитная карта - тоже документ. Заметь: приведенную выше информацию можно с успехом применить в корыстных целях, однако я надеюсь, что этого не произойдет. Ты останешься жестянщиком, который знает, что такое честь и честно заработанные деньги. В следующей статье расскажу о том, как изготовить устройство для чтения/записи магнитных карт, так что готовь паяльник - он тебе понадобится.

Честь имею! Ne01eX.

На мыло отвечали Dr. Klouniz vs SkyWriter

E-МЫЛО

(spec@real.xakep.ru)

ELENA NAMMA5895720@SUBSCRIBE.RU ЛУЧШЕЙ ПОДРУГЕ

Привет, Ленка!
Прости за молчание. Ты все равно моя лучшая подружка. У тебя самое доброе сердце, и поэтому ты все поймешь. Ленка, я попала...

Все началось с того, что шеф решил отметить своей "крысе" день рождения. Кому он это поручил организовать, ты, конечно, уже поняла. Я, как умная Маша, порылась в спаме и нашла несколько "праздничных" компаний. Всех обзвонить не успела...

Какие там мужчины! Когда увидишь, все поймешь сама (www.xakep.ru). Фотографии на странице "О нас". Кстати, отгадай, какой из них теперь мой (хотя не факт). Ты знаешь, я даже не обратила внимание на Диму Маликова и "Шоколадного зайца", хотя меня с ними и познакомили. Шефу и "крысе" все понравилось настолько, что теперь у меня есть лишних двести баксов за организацию! Погуляем, когда смогу оторваться от своего мужчины. Скорее посмотри и сообщи (ты точно должна отгадать).

Целую.

Пока.

ОТВЕТ:

Привет! Ой, правда, какие мужчины, особенно на странице www.xakep.ru/articles/common/info.asp. Я тоже, когда мне что-нибудь нужно, люблю порыться на радиоактивных помойках, в спаме, в унитазах и прочих местах, где дают дешево, много и сердито. Получается все равно плохо, поэтому подруги дразнят меня щитницей и помоешницей, но я не унываю. Чмоки, солнышко, обязательно пропьем твои двести баксов, когда начальник поймет, откуда у спаммеров берутся "професиАнальные актеры", "качественные локалЕзации" и другие выгодные предложения. А заодно и мужчинами померяемся.

Целую.

Везде (фу какая гадость, аж самому противно :-).

P.S. Конечно, ссылка в этом письме шла на другой, непроверенный сайт, но я побоялся его открывать и заменил ссылку на надежный ресурс :).

Я [KYKAPEKY@POCHTA.RU] КОРСАР@FTP.POCHTA.RU ПОЧТА ЖУРНАЛА (SPEC@REAL.XAKEP.RU)

Здравствуйте!
У меня такой вопрос... Множество моих знакомых используют антивирус-PandaAntivirus, который является еще и файрволом (вроде бы как) - все нахваливают... Про него в вашем журнале я не видел ни одного упоминания... Разъясните, пожалуйста, что это за антивирус и стоит ли его устанавливать. У меня установлены Dr.Web и Agnitum Outpost. Может ли PandaAntivirus их заменить? Или это все реклама? Заранее благодарен за ответ.

ОТВЕТ:

Привет.

Думать можно сколько угодно, но старая Панда, помню, не умела сканировать потоки в NTFS , что не есть гуд. Из антивирусов я предпочитал DrWeb SpIDer guard как монитор (он занимает около 900 Кб, совершенно не тормозит), однако детище Игоря Данилова пропускает некоторые вирусы. Как сканер я использую KAV и свои прямые руки. Еще, говорят, хорош NOD32. Как файрвол у меня стоит Agnitum Outpost, и это отлично. Вывод: что ставить, решать тебе. Я бы себе пушистого медведя не поставил (а я и пушистому Данилову вкупе с чемпионом по забросу якорей на длительность Касперскому предпочитаю своевременные обновления и прямые руки – прим. SkyWriter'а).

ДИМАН МАЛЫШЕВ MR.MYSKYLINA@MAIL.RU ВЗЛОМ МЫЛА :)

Здорово, коллеги-хакеры. Я, можно сказать, начинающий хакер.
У меня к вам просьбы:
1. напишите, пожалуйста, сайты или эл. книги (где скачать), как взломать мЫло.
2. где найти инструкции или эл. книги по программированию.

Заранее благодарю. Диман (Sn@iper).

ОТВЕТ:

Здорово!

Взломать мыло нынче трудно из-за дурацких "замков от детей". Просто на Западе дети часто открывали пузыри (ga-ga, везде дети открывают пузыри...) с мылом (с жидким, естественно), пили его, заливали себе в другие естественные отверстия и пачкали мебель. Поэтому изобрели специальную крышку: ее надо не просто крутить, а сначала надавить, потом вращать. И не надо никакого взлома. А насчет книг – в декабрьском номере в разделе "Подарки" посмотри, пожалуйста. Также – vr-online.ru, delphi.mastak.ru, delphikingdom.com, rsdn.ru.

АНОНИМ
ТЕМА НЕ РАСКРЫТА ;)

Привет, спецы. Я не пойму, что у вас произошло с последним номером. Вы что, корректора на тот свет отправили? Столько опечаток я еще в жизни не видел. Ладно бы в январе такой номер вышел. Я бы понял :)). А то ведь до нового года еще далеко. А так журнал отличный. Уже полгода читаю. Молодцы.

ОТВЕТ:

Да, до Нового года еще далеко... 11 месяцев как-никак :-). И ты угадал: Горл убил корректора и съел потому, что он имел наглость не знать, что такое брейкпойнт. Теперь мы все его боимся (не брейкпоинта - Горла :-). Так что теперь с орфографией, грамматикой и прочим мы не дружим. Не стреляйте: пишем, как умеем. Хотя стоп: если выгорит, то наймем профессиАнальных актеров. Они и корректировать будут, и на английский переводить. Как умные Маши... Только в спаме пороемся, погоди :). О! www.xyligan.ru/magazine/xy/030/056/1.asp

KARMA-COMA KARMA-COMA@MAIL.RU
ПОЧТА ЖУРНАЛА (SPEC@REAL.XAKEP.RU)

Hello spec! Привет, глубоко-глубокоуважаемый журнал СПЕЦ Х. Всегда любил Ваши спецвыпуски именно за пристальное внимание какой либо теме. И обсасывание ее со свех сторон понятным языком. Я не извращ. Просто так само написалось :) И это здорово! Читать спецвыпуски интересней обычного Х (в котором много кала для нубов). Но иногда при чтении Спеца сталкиваешься с тем, что Вы даете какие-то аббревиатуры без расшифровки. Я знаю много сокращений от IBM и RTFM до lol и FSOL. Но иногда попадается что-то новое. Например, в номере "Атака на вынь" в статье "Зло и ослик" встретил ATL. Не знал, что за зверь. Пришлось лезть в Сеть искать.

Вам из вредности не скажу, сами теперь поройтесь. А на будущее прошу все аббревиатуры, которые вы будете печатать на страницах журнала, расшифровывать хотя бы раз в скобочках, как только встретится впервые в текстке. Будет очень удобно. Имхо. Спасибо за внимание. Пис. Ваша Карма.

ОТВЕТ:

Hello, Карма!

Второй раз привет тебе! Безумно люблю наших читателей именно за "пристальное внимание какой либо теме. И обсасывание ее со свех сторон понятным языком". И все это они шлют нам в письмах... Мы тоже не извраты, но читать приходится. И кал от нубов, и другие разные анализы. Иногда себя лабораторией в поликлинике чувствуешь...

А по поводу аббревиатур - так мы тоже не валенком строганы, знаем и МТС, и ВЛКСМ, и КПСС... А вот еще прикольная: СССР - семья, сортир, столовая, работа. А расшифровки ДОК КЭУ МВО №27 не встретишь ни в одной сети, сколько туда не лазь. Хотя их как минимум уже 27 штук. А я ведь там проработал полгода...

Ну-ка, глянем-ка в словарь аббревиатур... Есть ласковые - МИБ, ЛАГ, НИИ; есть устрашающие - КГБ, МБРР, НИОКР; есть просто неприличные - ЭКОСОС, ЯМР, ОСРАК, ОСНАА. А сокращения?! Вот "НИИВТОРСЫРЧЕРМЕТБРЕДБРАКМРАКСНАБСТЫДСБЫТЗАГРАНПОСТАВКА". Каково? Но, если серьезно, обещаем исправиться. Обещаем выпустить отдельным изданием словарь наших сокращений, а со следующего номера подумываем печатать в конце алфавит (с транскрипцией!), чтобы легче было читать аббревиатуры в "текстке", а потом учиться их правильно произносить. И все это - не залезая в Сеть!

А кстати, Клуниз плакал все выходные, потом напился с горя и не пришел на работу :-(- расскажи ему про ATL, а?... Ну, пожаааалуйста!

Имхо. Пис. Слушай рок, Карма. Твои Спецы.

рис. Константин Комардин

Niro (niro@real.xakep.ru)

СЛУГА

Вчера посмотрел "Ночной Дозор". Я, конечно, его читал. И "Дневной" тоже, а вот неделю назад купил "Сумеречный", только руки все никак... И, думаю, вряд ли дойдут.

Сложно все это рассказать.

Но придется.

Помнишь саму книгу: тема равновесия сил и контроля проходит через нее красной нитью. Философия, не самая понятная, да и не очень толковая, по сути, ничего не объясняющая, а лишь говорящая тебе (если ты, конечно, склонен верить подобной чуши): "Помни: где-то есть люди, защищающие тебя от зла, мрака и боли; люди, принимающие на себя удар; люди, живущие этим и ради этого".

Где-то есть люди...

Наверное, я начинаю не с того. Опережаю события. Хотя нет ничего хуже хронологии, слепого следования времени и фактам. Разве не интересны, разве не интригуют голливудские фильмы, сбивающиеся на прошлое, на будущее, на параллельные и виртуальные сюжетные линии? Еще как! Примеров масса - "Подозрительные лица", "Шоу Трумэна", "21 грамм"... Я всегда любил фильмы, в которых понятно, о чем идет речь, только в последние двадцать секунд. Вот только весь фильм чувствуешь себя полным идиотом...

Я как никогда готов к тому, чтобы попытаться насытить свой дневник сюжетной линией, перекрестиями прицелов и словесных баталий, чудесами и буднями - готов претендовать на нечто напоминающее "Ночной дозор". И при этом очень не хочу оказаться вторичным, вялым, безвкусным подражателем.

Не хочу оказаться выжатым и вновь опущенным в воду пакетиком чая.

Хотя... Как сказать, как сказать. Выжатым... Пожалуй, этого у меня не получится, даже если я очень постараюсь.

Из моего пакетика всегда будет неплохая заварка.

Вы когда-нибудь слышали такой термин - "писательский скилл"? Полгода назад, когда я пытался выложить на просторы интернета свои произведения - первые, неокрепшие, зеленые рассказы - один из читателей сравнил меня с зубрами, с классиками жанра и ткнул пальцем в тексты, посчитав, что (цитирую) "...у того же Лукьяненко писательский скилл повыше будет".

Был. Теперь уже нет. За это время я поднабрал в свои мозги такую кучу дерьма, что построить многоэтажную фразу, которая в начале вызовет у читателя смех, а в конце слезы, для меня давно не проблема. Не верите? Перечитайте эту страницу - разве я ошибаюсь?!

Так... Увлекся, как обычно. Теперь надо вернуться к тому, с чего начал. Вот блин, всегда упирался рогом в начало - первые строки порой настолько бессмысленны, что поражаешься, откуда берется все остальное; никакой логики. На дворе март, а в окна врывается "свежий осенний ветер". Чушь.

Давай так. Я просто буду рассказывать. О себе, о своей жизни. О том, как все произошло и почему я никогда не прочитаю "Сумеречный Дозор".

Помнишь, я начал с того, что где-то есть люди? Этакие Люди с Большой Буквы, эти, будь они неладны, Иные, или Еще Какие-нибудь - "а вы, друзья, как ни садитесь, все ж в музыканты не годитесь"?

Так вот. Никого нет. НИ-КО-ГО. Нет и никогда не было. И даже не надейтесь. Не придут. Не спасут. Не помогут.

Вранье все это. Чистой воды. "Братья Гримм все придумали..."

А ведь знаешь - так жаль... Честно. Аж зубы скрипят, настолько нелепо все...

Ладно. Вперед. А ты потом сам решай. Я уже решил насчет "Сумеречного Дозора", что не буду читать. Простите уж, господин Лукьяненко.

То, о чем мне хочется рассказать, случилось не так уж давно, три дня назад. Конца и края не видно тому кошмару, который поселился в душе. А начало всему положило мое желание заработать. Банальное начало. Когда человек понимает, что ему не хватает на жизнь, это самое понимание повергает его в шок. Правда, перед этим он пытается охватить взглядом окружающее его пространство и подумать, где же можно поднять деньги с земли. Вот тогда и выясняется, что он никому не нужен и что, соответственно, никто не собирается ему платить. Знаний и умений недостаточно, чтобы предложить себя на рынке услуг, наглости тоже пока не набирал, воровать не научился, на второе образование времени нет, а работать на бензоколонке или автостоянке глупо, ибо можно растерять и все то, что пока еще осталось в голове.

Когда я сам посмотрел на свою жизнь со стороны и понял, что мне не светит доставать деньги из любого кармана брюк не задумываясь о том, откуда же взять еще, этот самый шок поразил меня в самое сердце; однако не все было так плохо, как я описывал несколькими строками выше. Кое-что я все-таки умел.

Я умел УЧИТЬСЯ.

И мне пришло в голову, что я просто обязан научиться чему-нибудь. А когда мой взгляд упал на компьютер, до сегодняшнего дня бестолково стоявший на столе, то понял, что моя будущая работа будет связана именно с ним.

Я опустился перед компом на стул, посмотрел на свое отражение в темном экране монитора и спросил сам у себя - а чем бы я хотел заниматься? Что такое может увлечь меня и принести не только радость от работы, но и реальный доход? Что может заставить меня вновь уважать самого себя?

Мне кажется, что некоторые люди, читающие сейчас эти строки, уже догадались, что же именно я выбрал. Нет, не идиотские затеи типа кражи виртуальных денег, взлома далеких и таинственных систем, похищения паролей и прав доступа - ничего похожего. Мне вообще всегда претило хакерское искусство, да и за искусство-то я его не считал, так - баловство одно. Сидишь за компьютером, пьешь пиво, разглядываешь строки в консоли, делаешь какие-то выводы, радуешься каждому открытому порту... Чушь. Хотелось чего-то более реального, чего-то, что можно, грубо говоря, подержать в руках.

И я нашел. Все оказалось просто.

Программирование.

Я, честно говоря, сам сейчас не могу объяснить, как же так получилось, что я обратил внимание на написание баз данных. Вроде бы и реального приложения подобных умений в моей жизни никогда не было, но, тем не менее, я кивнул сам себе, вышел на улицу и через пару часов вернулся домой с несколькими книгами под мышкой.

Хочешь понять, как работает программа дяди Борланда, - создай форму, кинь туда кнопку, напиши процедуру Close, запусти, нажми кнопку, и форма закроется.

Я чувствовал, что сумею; я был просто обязан научиться. Дело было не только в возможности обрести заработок: включились уже какие-то другие механизмы самореализации, необходимость подобного образования приятной дрожью отзывалась во всем теле...

Да и слово какое приятное - "Дельфи"...

Давай сразу договоримся - никаких споров о том, что лучше. Я выбрал Pascal, а ты можешь пробовать все остальное. Прими это за константу. Другого варианта на этих страницах не будет.

Итак, я разложил перед собой книги, нашел в интернете ряд ссылок и погрузился с головой в процесс постижения всего того, что скрывается за словом "Дельфи". Едва я прочитал первые несколько слов, я понял, что охватить все нахрапом вряд ли удастся. Короче, я застрял на всех этих циклах, операторах, процедурах, методах и объектах. Вроде бы все по отдельности было до ужаса логично и элементарно, но только попробуешь хоть что-то сделать, как выходит какое-то бессмысленное нагромождение - бесконечные ошибки при компиляции и тому подобная дребедень.

Конечно, я начинал, как и все, с банального "Hello, world". Конечно же, у меня все получилось - такие вещи просто не могут не получиться. Хочешь понять, как работает программа дяди Борланда, - создай форму, кинь туда кнопку, напиши процедуру Close, запусти, нажми кнопку, и форма закроется. Замечательная получается игрушка.

Может быть, кому-то все, что я сейчас рассказываю, может показаться неинтересным. Но вот, елки-палки, читают же люди эти порой глупые, а порой просто непонятные "ЖЖ", которые стали едва не новой формой сетевой религии. Почему вам не прочесть то, как какой-то паренек из глубинки взялся изучать Дельфи. Тем более все, что будет дальше, напрямую с этим связано.

Все дело в том, что слишком уж много места в нашей жизни отводится случаю, хотя множество философов, писателей-классиков и еще куча всяких людей, занятых исследованием базиса человеческого существования, считают, что это не так, что все очень закономерно вытекает одно из другого и при желании всегда можно выстроить очень и очень логичную цепочку между двумя, казалось бы, никак не пересекающимися событиями.

Вот и я - человек, живший до поры до времени по законам, изобретенным этими людьми, внезапно проникся ролью случая в истории. Так как я был поставлен перед фактом, скажу тебе: если уж случаев не быва- »

ет, то все, что произошло со мной, - стопроцентное исключение из этого правила.

Пытаясь сообразить, что же мне делать дальше, раз уж мозгов не хватает, я понял, что без наставника в этом деле не обойтись. И вот тогда я вспомнил про Ткачева.

Про Мишку Ткачева, с которым мы учились в одном классе. Он после школы выбрал информатику, а я - литературу и журналистику (как выяснилось, зря: стать знаменитым в этой области практически невозможно, если только у тебя в кармане кучи баксов... Вот блин, опять все деньги, деньги!..) Безусловно, Мишка в то время понятия не имел, правильный ли шаг он делает. Что такое одна тысяча девятьсот девяностый год? Никаких "Пентиумов". "Винду" напишут еще только через пару лет, программы кропают на каком-то Бейсике - короче, болото. И тем не менее, выбор он сделал - и, как выяснилось, правильный.

Мало какая отрасль рванула за столь короткие сроки так далеко вперед. Пожалуй, космонавтика и та тормознула на многие годы с развалом всего и вся. А вот компьютеры, программирование и прочие атрибуты хай-тек могут дать фору кому и чему угодно. Мишка удачно устроился в этой нише, освоив все тот же Паскаль, пропитавшись им насквозь, словно верой. Иногда казалось, что он даже говорит стараясь строить фразы исходя из законов не русского языка, а алгоритмического.

Человек своего рода свихнувшийся - так можно было бы охарактеризовать его, но все оказалось гораздо хуже. Мне очень неприятно об этом вспоминать, но факт остается фактом - у него было еще одно увлечение, которое сыграло большую роль во всем происходящем. Не могу не упомянуть об этом.

Имя этому увлечению - "огненная вода".

Сколько я его помню, он постоянно пил. В смысле, помню-то я его со школы, тут я немного погрешил против истины. Он начал пить года, наверное, три назад - и никто из его друзей не мог никогда толком сказать,

Я подумал, что хорошо сделал, что не выставил на стол вторые два литра.

что же послужило причиной этого падения. Не было в его жизни ни горя, ни смертей, ни неразделенной любви...

Почему-то вдруг подумалось: а что я знаю про него по-настоящему? Может, это был единственный факт, о котором я знал, но я склонен думать иначе.

Короче, вспомнил я про Мишку и тут же понял, что ни к кому другому за помощью идти смысла нет, ведь лучше Ткачева объяснить чего-нибудь о компьютере и программах не мог никто. Вот только надо было застать его трезвым, лучше с утра - пусть даже больной, но еще с ясным рассудком он все-таки лучше, нежели с заплетающимся языком и перепутанными пальцами, не попадающими в клавиши. Часы к тому времени показывали уже послеобеденное время, я покачал головой, прикинул свои финансы и решил, что завтра утром возьму немного пива и буду использовать его в качестве приманки - если захочет полечить голову, будет более сговорчивым.

Такой подход может показаться жестоким, но друзьями закадычными мы с Ткачевым никогда не были, да и я знал о том, что жмот он порядочный: если не захочет делиться информацией, то клещами из него ничего не вытащишь. Поэтому, как говорили иезуиты, цель оправдывает средства.

По деньгам - хватало бутылок на пять. Если разливное, то выигрыш получался приличный. Приняв решение, я успокоился, запустил новую игрушку и оторвался по полной...

На утро я первым делом рванул на "точку", взял четыре литра свежего "Жигулевского" и несколько пакетиков "Кириешек", помчался к Ткачеву, чувствуя, как в кармане куртки бьется и просится наружу мой винчестер - мало ли сколько инфы сумею стрясти с Мишки?

Насчет его утреннего состояния я оказался прав: дверь мне открыл человек, обозленный на все и вся, его мутный взгляд блуждал где-то поверх моей головы, на трясущихся пальцах рук звенели ключи, когда он ковырялся в замках дверей за моей спиной. В воздухе стоял запах перегара, корейских салатов и еще чего-то приторно-сладкого. Я не мог сразу понять, но когда увидел возле вешалки женские туфли, голубую блузку на кресле и ткнулся взглядом в закрытую дверь спальни, то понял, что это какие-то дешевые духи.

- Чего тебе? - спросил Ткачев, впустив меня и глядя по-прежнему куда-то мимо. В ответ я молча поднял перед его глазами пакет. И хотя угадать его содержимое простому человеку с первого мгновенья было практически невозможно, Мишка выпрямился, как бамбуковый прут, его глаза сверкнули, он схватил меня за руку повыше кисти и быстро завел на кухню.

Я почувствовал, что с первых же шагов теряю инициативу - Ткачев видел во мне просто "Скорую помощь". Я попытался вырваться - безрезультатно. На столе оказались две большие пивные кружки. Ловким движением Мишка выхватил у меня пакет, открутил пробку с одной из пластиковых бутылок, опрокинул ее, пена и пиво рванули в кружку...

Алкоголизм ведь все-таки болезнь. Если понимаешь, что этот человек болен, начинаешь относиться к нему несколько по-иному. С жалостью, что ли. С долей понимания, что не сам он уже рвет пробку с бутылки, что это больной организм приказывает ему, что сейчас делать, какими глотками и какую жидкость пить. Мозги, уставшие и воспаленные, требуют увеличить дозу этилового спирта в метаболизме. Никакой прихоти, никакого управления своими желаниями - деваться уже некуда. Был человек - и нету.

Было горько смотреть на то, как он ждет, когда наполнится кружка. Был бы, наверное, поздоровее, наклонил бы ее, лил медленно, ждал, когда опадет пена. Тогда все было не так: плюхнул бутылку, белая шапка рванулась на стол, он схватил кружку обеими руками и жадно прильнул к ней, не замечая, как пена плывет по подбородку и капает уже на колени.

В тот момент он был похож на человека, только что вышедшего из пустыни. На какое-то мгновенье мне показалось, что я пришел не по адресу - уж очень плохо он выглядел. Но с каждым глотком к нему возвращалось все то человеческое, что вечером накануне он похоронил в стакане. Он поставил кружку на стол, протянул руку за спину, ловким движением ухватился за кончик полотенца и утерся.

Из спальни что-то пробурчали.

- Не обращай внимания, - сказал Ткачев и налил себе вторую, полную. На этот раз он был чуточку осторожнее, расплескал гораздо меньше. - Спасибо тебе, - вставил он пару слов между глотками.

Мне ничего не оставалось, кроме как кивнуть и ждать, когда и третья кружка уляжется на свое место в желудке. Ткачев поставил ее на стол, сжал губы и зажмурился, прислушиваясь к тому, что происходило сейчас в его организме.

- Неправильно проведенная опохмелка ведет к запою, - сказал он, не открывая глаз. - Стоп!

И взглянул на меня. Я поразился перемене, которая случилась с ним за последние несколько минут. Передо мной стоял абсолютно вменяемый, спокойный, лишь слегка покачивающийся человек с осмысленным взглядом.

- Так ты зачем пришел? - спросил Мишка, видя мое замешательство. - Ведь не затем же, чтоб меня пивом напоить?

Конечно же, нет. Я смотрел на него не в силах произнести ни слова. Из спальни снова что-то крикнули. Противный, визгливый девчоночий голос. Ткачев указал мне на табуретку, после чего пошел в комнату. Там что-то прошуршало, скрипнула дверь: похоже, в спальне кто-то одевался. Через несколько минут стук каблуков подтвердил мои догадки, девушка вышла в коридор, взглянула на меня, хмыкнула... Я даже не потрудился ее рассмотреть - прекрасно понимаю Ткачева, он и сам был неприятно удивлен, увидев ее, когда стал трезвым. Тем временем девица подошла ко мне, взяла со стола недопитую бутылку, обеими руками поднесла ее ко рту и осушила.

Я подумал, что хорошо сделал, что не выставил на стол вторые два литра. Ткачев подошел, пихнул даму в спину и выставил за дверь. Потом высыпал в ладонь горсть сухариков и сквозь хруст предложил мне пройти в комнату; я подчинился, подхватив с пола пакет. Сам Мишка взял кружки.

Комната разительно отличалась от кухни, в которой был жуткий бардак. Похоже, к компьютеру и всему, что с ним связано, Ткачев относился с благоговением.

Я увидел два больших книжных шкафа с множеством специальной литературы; несколько стопок журналов, перевязанных бечевкой, по углам компьютерного стола; пара принтеров - довольно дорогой лазерный и струйная "дешевка"; несколько разобранных компьютерных корпусов со свисающими наружу проводами расставлены вдоль окна; на маленькой книжной полке возле двери - множество раскуроченных мобильных телефонов (вот уж не думал, что он и в них разбирается). Но больше всего меня поразили книги.

Почти все они были на английском языке. Куча специальных зданий от самого дядюшки Борланда; какие-то пособия, самоучители, справочники команд. Когда я увидел все это, то первое, что понял, - мне никогда не овладеть программированием. Ткачев понял, что именно поразило меня, похлопал по плечу и сказал:

- Впечатляет? Меня тоже.

- Неужели ты все это читаешь в оригинале? - спросил я, не в силах поверить.

- А то... - он развел руками; несколько сухариков упали на пол, он тут же кинулся их поднимать со словами "Пока не раздавили..."

Я присел на диван, собрался с духом и объяснил Ткачеву, зачем пришел. Тот выслушал меня, нахмурив брови и тщательно стал пережевывать "Кириешки". Трудно было понять, нравится ли ему то, что я говорю, готов ли он поделиться своими знаниями со мной или нет. Он медленно шевелил челюстями, изредка откидывая голову на спинку кресла и прикрывая глаза. Чувствовалось, что ему стало несравнимо лучше после пива. Иногда он кидал взгляд на пакет со второй пластиковой бутылкой, но предложений открыть ее пока не поступало.

Тем временем я выговорился; изложение проблемы поиска финансов заняло у меня много времени, а уж попытки внятно объяснить, почему я собрался зарабатывать деньги при помощи Delphi, было вообще сложно.

Ближе к концу разговора Ткачев зевнул широко и длинно. Я было испугался, что его сейчас разморит, он заснет и из моей затеи ничего не выйдет, но он внезапно наклонился ко мне и сказал:

- Попробуем...

После чего достал из пакета вторую бутылку, открыл ее и медленно и аккуратно налил себе и мне по кружке.

Запустив программу, он пригладил волосы и, судя по взгляду, предложил мне придвинуться поближе. Я принес из кухни табуретку, присел сбоку.

- Все просто, - сказал Мишка. - Главное - чтобы ты умел логически мыслить. Не факт, что "Дельфи" будет указывать тебе на ошибки сама. Если ты не в состоянии написать простейший алгоритм, если ты будешь путаться в циклах, потому что не увидишь в них элементарной логики, то путь, который ты выбрал, явно не для тебя.

Я кивнул, понимая все это не хуже его самого.

- Поэтому слушай и запоминай. Базы данных начинаются всегда с самого простого и самого главного - с цели. Что именно и как ты собираешься упорядочить; зачем тебе все это и как потом ты вытащишь из своей базы нужные данные. Короче, главное - правильно скомпоновать таблицы и установить между ними связи...

И он принялся мне объяснять все с самых азов. Поначалу я просто слушал, потом принялся записывать в блокнотик. Ткачев, потихоньку отхлебывая из кружки пиво с уже опавшей пеной, постепенно накачивал меня информацией.

Как много зависит от того, каким языком и насколько понятно человек объясняет тебе решение проблемы! У меня через несколько минут общения с Мишкой сложилось впечатление, что он только и создан для того, чтобы читать лекции по информатике, программированию и еще по многих дисциплинам, имеющим отношение к точным наукам. Настолько просто и удобоваримо мы с ним продирались через дебри Паскаля, что я даже не заметил, как пролетело около двух часов. Только количество пива в бутылке отмечало ход времени - оно постепенно перекочевало из пластиковой емкости в Мишку, сделав его еще более разговорчивым; вот только речь его стала какой-то вязкой, неуверенной, лишь пальцы все так же быстро порхали на клавиатуре да мышка пока ни разу не промахнулась...

Постепенно я начинал понимать, и передо мной все более четко вырисовывались все перспективы того дела, за которое я решил взяться. Я уже видел людей в строгих костюмах, несущих мне чеки за программы, написанные для их мегакорпораций, директоров, предлагающих мне высокие посты в их компаниях, короче, пока все на экране делалось руками Ткачева, жизнь казалась радужной.

- А теперь попробуй сам, - внезапно сказал он мне чуть ли не посредине своей очередной фразы. - Берем в руки приборчик, тычем пальчиками в кнопки, короче - работаем. А я пойду на кухню, сварганю что-нибудь... Вроде яичницы.

Я занял его место, взглянул в расчерченную на бумажке схему и принялся набрасывать тренировочную базу кое-как, едва ли не на коленке. Время от времени я прислушивался к тому, что происходит на кухне, - какое-то шипение, стук кастрюли, шум льющейся воды; Мишка вовсю хозяйничал там, полностью отошедший от похмелья.

Примерно через полчаса я понял, что овладел некими начальными навыками. Я делал таблицы, присоединял их к проекту, подключал сетки, просматривал данные, компилировал, запускал - все работало. Правда, я понимал, что работает все пока по одной причине, - исключительно из-за простоты. Сделать ошибку в том, что я построил, было невозможно.

Это меня и радовало, и пугало одновременно. Я чувствовал, что мое желание работать пока ничем не подкреплено. Так, мелочи какие-то.

Ткнувшись пару раз в незнакомые мне функции, я все-таки сумел сделать какую-то ошибку, развел руками и крикнул на кухню:

- Мишка!

И тут же понял, что оттуда не доносится ни звука, только потягивает чем-то горелым. Я выскочил из-за компьютера и бросился на кухню.

Картина была довольно типичная, прямо-таки из бывших совдеповских "чернушных" фильмов. Мишка спал навалившись грудью на стол; на плите благополучно начинала гореть яичница, а рядом со спящим Ткачевым стояла полупустая бутылка водки и стакан.

- Эх, елки-палки! - я ринулся к плите, спасая комнату от вони и возможного пожара. - Нашел свою заначку!

Было похоже, что наше обучение на сегодня закончилось. Я опустился в кресло, крутнулся в нем пару раз, разглядывая квартиру Ткачева и вслушиваясь в его громкое сопение. Жалко было бросать все едва начав; я попытался продолжить делать то, что начал, опять напоролся на какие-то непроходимые ошибки и бросил. А потом мне вдруг пришло в голову, что неплохо было бы попробовать делать хоть что-нибудь слепо копируя. Ведь должны же быть у Ткачева на компьютере какие-то свои собственные разработки, глядя на которые можно разобраться во многом, а что не получится - так он ведь не будет вечно пьяным!

Я вытащил винчестер из своей куртки и полез под стол. Наладив все, что нужно, я принялся изучать содержимое Мишкиного компьютера и довольно быстро нашел его рабочие документы, папки с исходными кодами, нереализованными проектами, какими-то наработками и просто непонятно с чем. Места на моем диске было более чем достаточно; я выделил все, что посчитал нужным, и запустил копирование.

Информация принялась перекачиваться на мой винчестер. Я внимательно следил за ползущей синей полоской и быстро сменяющими друг друга процентами и думал о Ткачеве. Насколько неприятно было видеть все это: падение человека, с которым ты учился вместе не один год и никогда не видел в нем никаких предпосылок к подобному развитию событий. Мы сидели с ним на одном ряду через две парты, всегда писали один вариант, и на контрольных по математике он неизменно решал всему ряду и мне в том числе. Учителя видели в нем будущее светило точных наук, поэтому никто не удивился, когда он выказал желание поступать на информатику. Его благословили, написали кучу достойных характеристик, подготовили к вступительным экзаменам, и он благополучно прорвался в университет.

Я уже видел людей в строгих костюмах, несущих мне чеки за программы.

Помню, после первого курса классный руководитель собрал нас всех вместе - в первый и последний раз. Мишка тогда пришел со своей будущей женой (женился он потом быстро, через месяц, так же быстро и развелся, никому не объяснив причины - детей у них не появилось, вечные скандалы, грызлись по мелочам). Он всех нас грузил своими познаниями в кибернетике, логике и еще куче всяких дисциплин, которые они изучали; наши девчонки смотрели ему в рот и с ненавистью обсуждали за глаза его невесту. Его все любили...

Куда все подевалось? Что сталось с ним, что развернуло его к жизни на сто восемьдесят градусов? Такие вещи не происходят беспричинно: толчок приходит либо извне, либо изнутри. Каждый из похожих на Ткачева может поведать душещипательную историю о том, как все это случилось. Не стоит верить их словам буквально, просто посмотри им в глаза - там есть ответ.

Вот только почему-то в глазах Мишки этот ответ можно было прочитать только с большим и очень большим трудом. Я поймал себя на том, что не отрываясь смотрю на него, пытаясь проникнуть в его тяжелый пьяный сон, почувствовать его, понять...

Компьютер пикнул, сигнализируя, что обмен закончен. Я вздрогнул и вышел из оцепенения, навеянного размышлениями о жизни. Мишка что-то пробурчал, перевернулся на другой бок и потянул плед, стащив его с ног на голову. Я снова нырнул под стол, восстановил статус quo, сунул изрядно нагревшийся винчестер в куртку; потом задумался на мгновение и отправился на кухню.

Остатки водки я вылил в раковину, пустив сильную струю воды, чтобы утопить в стоке даже запах. Сполоснул стакан, прибрался немного на столе, собрал пивные бутылки в углу в большой целлофановый пакет; мне почему-то захотелось отблагодарить его за его знания, за ту информацию, что я сейчас переписал себе и которая, возможно, сделает из меня достойного программиста. Подойдя к двери, я последний раз оглянулся на Мишку, крепко спавшего на диване, вышел на площадку с двумя мусорными пакетами и захлопнул дверь.

Придя домой, я не торопясь разделся, зашел в комнату, положил винчестер на стол посреди учебников и дисков, сел в кресло и задумался. »

Черт его знает, что меня там посетило, уже и не вспомню, но просидел я довольно долго. И это несмотря на то, что руки у меня потихоньку чесались, что мне не терпелось поскорее начать, я сдерживал себя по непонятным причинам.

По непонятным тогда... Сейчас-то я прекрасно понимаю себя. Мне было сложно вернуться из мишкиного мира в свой, в нормальный, упорядоченный, ТРЕЗВЫЙ мир. И мне было стыдно, что я здесь, а он - там.

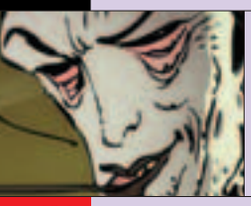
Правда, это быстро прошло. Примерно за полчаса. Несколько вздохов, пару раз протер глаза, хмыкнул, покачал головой - и прошло. Как рукой сняло. Было ощущение, что я принял какую-то таблетку, действие которой потихоньку вытравило из меня всю эту чернуху. Я встал, прошелся по комнате, присоединил винчестер и принялся просматривать все то, что скопировал у Ткачева.

Информации получилось очень и очень много. Около тридцати проектов, в них десятки, сотни юнитов, модулей и прочей дребедени, которая носила гордое имя "Дельфи". Кое-что можно было понять сразу по названиям проектов, но основная масса, похоже, была известна лишь Ткачеву, ибо нумеровалась в каком-то хитром порядке цифрами и буквами.

- Черт ногу сломит, - бурчал я, глядя на все это. Часть скомпилированных проектов можно было запустить - я делал это глядя на то, с какой легкостью распахиваются передо мной хранилища данных совершенно разных размеров и направлений. - Ну вот это, пожалуй, я пойму - какая-то фирма по продаже сигарет... А вот это, похоже, чья-то библиотека...

Мне оставалось просмотреть всего три проекта на предмет чего-нибудь интересного для меня. Один из них был большим каталогом фильмов, второй - чем-то вроде бухгалтерской программы, правда, я так и не понял, в чем суть и что именно там считалось. А вот третий...

Это был список людей. Ничем особенным не объединенных, никаких общих черт у них я не нашел. Просто большой список, даже не большой, а огромный. Фамилии, имена, адреса, даты рождений, знаки Зодиака, еще



Глаза шарили по строкам, разглядывая окошки поиска, меню, "мышка" носилась по экрану как угорелая...

кое-какие непонятные графы с обилием цифр. Напротив всех строчек стояли галочки; меня все это заинтересовало вначале постольку, поскольку в коде этой базы могли быть интересные процедуры по поиску и сортировке (Ткачев всегда отличался нестандартным подходом к своей работе, стоило ожидать и здесь каких-то программерских хитростей и красивых действий).

Я открыл редактор кода, прошелся по нему взглядом, выхватил то, что уже понимал, - обработчики нажатий кнопок, некоторые простые циклы... И постепенно понял, что эта база - пожалуй, самое сложное и непонятное из всех проектов, сохраненных на компьютере Мишки. Масса обращений в никуда, к таким виртуальным вещам, как Зодиак и ему подобные. Как все это могло работать, трудно было сказать.

Я проверил некоторые непонятные моменты по учебникам и не нашел в них ничего похожего. Ни одна из процедур, обращающихся к полям с цифровыми группами, к полям с таинственными значками, не была определена в книгах. Складывалось впечатление, что Мишка пользовался какими-то недокументированными возможностями Delphi, не описанными нигде: ни в книгах, ни в справочной системе, ни на сайтах поддержки. Возможно ли, что он сам создал какие-то средства разработки? Вполне. Он был человеком очень и очень одаренным, способным на многое - я бы не удивился ничему, в том числе и такому повороту событий.

Поначалу я хотел бросить изучение этого хитрого проекта, потому что моего ума в настоящий момент едва хватало на простые вещи, описанные в каждом справочнике, - разбираться в неведомых командах, рожденных явно не без помощи бутылки, у меня не было ни сил, ни желания, ни времени. Выкинув из головы это необычное творение Ткачева, я заставил себя заниматься используя то, что он рекомендовал мне для начала - "Библию Delphi" Михаила Фленова. Уж очень он нахваливал мне своего тезку и его способ преподносить информацию. По этой книге я уже через час сделал свой телефонный справочник, выдергивал из него информацию, кропая отчеты, писал свои собственные куски кода и был чертовски горд всем этим - вот только похвастаться было некому...

Тем временем за окном уже стемнело; глаза болели, желудок требовал порции калорий. Я решил на сегодня закончить. Себе я казался чуть ли не героем, победившим некое древнегреческое мифологическое чудовище, во мне просто бурлила сила программиста. Я был настолько уверен в себе, что собирался уже завтра дать в газету объявление о написании баз данных и пройтись по нескольким фирмам в городе в попытках найти достойную работу. Отсутствие диплома не пугало меня - в нашем теперешнем обществе это было далеко не самое главное. Я был уверен в том, что смогу произвести впечатление на тех, с кем придется разговаривать, а в качестве доказательства моих знаний и умений я мог привести Мишкины проекты...

Я решил сделать себе что-то вроде демонстрационного диска для тех, с кем мне придется общаться в ближайшее время для доказательства своих сил и умений. Для этого я собрался отобрать несколько самых интересных, на мой взгляд, проектов Ткачева и выписать их на болванку. Изучая их, я уже остановил свое внимание на пяти-шести - не считая последнего, который, как мне казалось, мог повредить моей репутации. Я был уверен, что, попадись мне действительно грамотный собеседник, я никогда в жизни не смогу объяснить, в чем смысл более чем половины кода этой базы данных; а рисковать таким образом я не мог.

Диск был готов через десять минут. Привод выдвинулся, я взял болванку в руки, приготовил ее на завтра, положив в коробочку поверх всей свой кучи софта. А через минуту понял, что неподвижно сижу в кресле и смотрю на запущенную ткачевскую базу. Ту самую, со знаками Зодиака.

Совершенно не помню, когда я ее включил. Просто она оказалась запущенной; я пялился в эти фамилии, даты рождения, весь этот табличный сюрреализм, и мне казалось, что сама программа чего-то хочет от меня. Так порой бывает - знаешь, что делать чего-то не стоит, но, тем не менее, делаешь, будто надеясь на что-то сверхъестественное. Эдакий "Format C:" - а вдруг не сработает? А он, сволочь, работает, форматирует, да еще как...

Так и я - глаза шарили по строкам, разглядывая окошки поиска, меню, "мышка" носилась по экрану как угорелая... Я сам не понимал, чего хочу от всего этого.

Борисов Сергей Степанович, двадцать второе января, Водолей. Это что за абракадабра... А вот Тимофеев Владимир Николаевич, четвертое марта, потом Рыбы, потом звездочка, человечек, потом список выпадает, а там по-латыни... Или по-гречески...

Я шептал все это себе под нос, одновременно прокручивая список людей, стараясь узнать, что же там, в конце. На две тысячи семьсот восемнадцатой строчке таблица кончилась. Не то чтобы у меня устал палец крутить колесико - но все-таки столько информации! Я вдруг подумал - а почему у Мишки на компьютере оказалась не пустая база для клиента, а заполненная? Это что, его личный проект, он сам следил за его заполнением, за всеми этими галочками и уродцами, за всякими словами, жутко звучащими на русском языке? Или кому-то было лень следить за целостностью базы, и этот "кто-то" доверил вбивание строк Ткачеву?

Тот еще вопрос. Действительно, почти три тысячи позиций в базе - это ведь не пять минут работы. Или Ткачев когда-то забросил все свои дела и только и занимался тем, что заносил сюда данные, либо постепенно, шаг за шагом, по две-три строчки в день, создавал все это нагромождение Водолеев и Скорпионов, расставляя, где надо, галочки...

Или галочки расставлял уже не он?

В общем, вся моя работа была забыта. Я уже не рвался писать свое резюме, не бомбил интернет в поисках ответов на вопрос о трудоустройстве - я только и думал обо всех этих Петровых, Борисовых, Михеевых и иже с ними, расставленных в таблице, исходя из таинственной логики.

Интересно, проснулся к тому времени Ткачев или нет? Подозревал ли он о том, что кто-то смотрит сейчас в те же строчки, что и он? Это осталось тайной для меня, хотя периодически я возвращался к этому вопросу; но это уже потом, когда я - лишь поверхностно! - сумел проникнуть в решение проблемы.

Колесико вертелось под пальцем туда-сюда, строки двигались вверх-вниз. Временами я залезал в меню, пытаясь сквозь череду модальных окон продраться туда, где совершалось главное действо. Ведь если люди вносились в таблицу - значит, это кому-нибудь нужно.

- Предположим, - сказал я сам себе, - эти галочки означают, что эти люди соответствуют какому-то условию. Или выполнили его. Или еще чего-нибудь... Да, или с ними что-то сделали - ну, я не знаю, подписались на журнал "Знаки зодиака", выдали гуманитарную помощь, еще какая-нибудь фигня!.. Но почти три тысячи человек!

Честно говоря, не знаю, что сбивало меня с толку больше: количество людей в списке или все эти значки, сопровождавшие каждого. Попытался войти в таблицу через редактирование - база тут же спросила у меня пароль.

- Ага, - сказал я. - Уже что-то.

Меня посетила мысль натравить на окно для ввода пароля какую-нибудь брутфорсовую софтину - почему-то казалось, что пароль сюда придумывал явно не Ткачев; скорее, автором доступа был сам хозяин базы. Вряд ли у него хватило фантазии на большее, нежели "qwerty" или "password", но не стоит недооценивать противника. Я пошарил в недрах компьютера, извлек необходимую программу, но что-то меня остановило.

И зашел в базу еще раз, но не через "Редактировать", а через "Добавить". И никакого пароля не появилось.

Добавлять строку в базу можно было свободно. Вот так и доверял неведомый хозяин Ткачеву - добавляй, кого скажу, а изменить ничего не можешь.

- И неужели Мишка ни разу не захотел подобрать ключик? - засомневался я. Потом устроил окну ввода пароля брутфорс, а сам отправился на кухню: желудок уже вступал с мозгом в открытый конфликт. Картошка с тушенкой успокоила и того, и другого. Я вернулся через сорок минут за компьютер в надежде получить пароль, но вместо этого увидел лишь неутешительный прогноз - подбор букво- или цифросочетания требовал около трех недель.

Не оставалось ничего, кроме как прекратить это бесполезное занятие. Я покачал головой, щелкнул пальцами от обиды и сделал то, о чем вы, наверное, давно уже подумали, только, может, не в том ключе, в котором оно пришло в голову мне.

Да, я решил добавить в базу еще одну строку. Еще одного пользователя.

И что самое интересное, никогда не мог потом объяснить, почему поступил так, как получилось. Обычно в таких случаях я вносил в регистрационную форму некоего Ивана Панкратова, виртуального персонажа, выдуманного мной еще в институте для авторизации своих первых статей. Я знал и дату рождения Ивана, и много чего еще - многочисленные форумы в Интернете тоже проглотили кучу моих "левых" панкратовских данных. Я с ним сроднился; я не мыслил себя без него.

Но почему-то в этот раз я внес в список свои настоящие данные. И проследил, чтобы все было абсолютно правильно - дата и место рождения, знак Зодиака и (обрати внимание на то, что требовалось в этом непонятном списке) любимое время года, любимую музыку, любимый запах, сексуальные пристрастия, потом наудачу выбрал из огромного количества непонятных рисуночков те, которые я посчитал "смайликами", симпатичную одноглазую мордашку. По списку с записями на латыни щелкнул не глядя - ни одного из тех слов я не знал, и никаких ассоциаций с русской речью они не вызывали.

В общей сложности получилось восемнадцать пунктов. Нажал "Сохранить" и "Выйти", посмотрел на результат своего труда. Строка с моими данными оказалась две тысячи семьсот девятнадцатой.

Естественно, галочку поставить не получилось - сетка была защищена от редактирования. Ну да и Бог с ней, с галочкой. Хватит на сегодня.

"Завтра будет трудный день", - решил я и собрался идти спать. База закрылась без лишних вопросов, сохранив мою строку в своих недрах. Монитор легонько мигнул и погас.

И в этот момент зазвонил телефон. На часах было почти одиннадцать часов, близилась полночь. Я вздрогнул и удивленно взглянул на брошенную на диван трубку. Звонок повторился.

- И понадобился же я кому-то... - пробормотал я и протянул к телефону руку.

...Я взял трубку и нажал кнопку. Свистящий, шипящий и шепелявящий Ткачев попробовал сказать мне в ответ "Алло" и, похоже, уронил телефон. В трубке раздался какой-то грохот, потом гудки. Я пожал плечами и подумал, что если ему будет нужно, он перезвонит. Наверняка перепутал день с ночью и хочет еще пива.

Телефон, конечно же, зазвонил вновь. Как-то громко и тревожно что-то шевельнулось во мне, говорило о том, что не так все просто и дело совсем не в пиве. Я снова нажал кнопку.

- Ты... Откуда ты там? - спросил Мишка.

- Я живу здесь, - ответил я на дурацкий вопрос. - Ты что, не знаешь, куда звонишь?

В те мгновенья моей стремительно утекающей прошлой жизни я все еще верил в бессмысленность и случайность этого звонка. Я до сих пор верю в это - хотя все уже случилось. Мне постоянно кажется, что жизни после звонка не существует. Пожалуй, каждый может вспомнить нечто в своей жизни, какую-то знаменательную точку отсчета, которая повергла тебя в шоковое состояние. Кто-то въехал на своих "Жигулях" в "Ланд Крузер"; кто-то узнал, что болен СПИДом... Дерьма много. И оно всегда когда-то начинается. Что-то из Стивена Кинга. По-моему, "дерьмо случается", так он говорил. А может, и не он. Да неважно все это.

- Я знаю, - сказал Ткачев (и я понял, что он достаточно трезв, просто, похоже, только что проснулся). - Это ты, это твой номер, и ты там.

- Где?

- В базе.

Я замолчал надолго. Мне даже вспоминается сейчас, что я сидел с открытым ртом, как даун. Сидел, слушал удары своего сердца и чувствовал, как во рту копится куча слюны, которую, если не проглотить вовремя, придется выпустить себе на рубашку.

- Где? - еще раз спросил я, прекрасно понимая, о чем идет речь.

- Ты знаешь, где, - сухо сказал Ткачев. - Чем ты ей насолил? Или тебя заказали? Вспоминай, только быстро - у тебя есть враги? Явные, тайные? Может, девчонка какая - ну, там, дала, а ты бросил? Короче, вспоминай всякую чушь! Только быстро!

Я не понимал ровным счетом ничего. Причем здесь база? Кому я насолил и зачем?

- Ткачев, - тихо спросил я. - У тебя все в порядке?

- Конечно же, нет, - бросил он в трубку. - Ты что, ничего не понимаешь?

- Нет, отчего же, я понимаю, - покачал я головой. - Я понимаю, что я в базе и что меня заказали... Ты вменяемый, Мишка?

Как-то громко и тревожно что-то шевельнулось во мне, говорило о том, что не так все просто и дело совсем не в пиве.

На том конце откашлялись, а потом зарядили в меня семиэтажным матом. Я отодвинул трубку от уха, выслушал эту тираду и хотел уже разъединиться, но Мишка вдруг заорал мне что-то, и я решил послушать дальше.

- Эй, там! - орал он, будто знал, что его маты я слушал на отлете. - Ау-у!!

- Я все еще здесь, - ответил я.

- Я знаю, это похоже на разговор двух дебилов...

Он замолчал, и я успел подумать, что это похоже на разговор одного дебила.

- И все-таки, - продолжил он. - Я думаю, есть смысл спросить тебя - ты знаешь, сколько лет я пью?

- Лет пять, - брякнул я, особо не задумываясь. Товарищу Сталину пару лет туда, пару лет сюда - не срок.

- Точно. Пять с половиной. Ты видел меня сегодня. Как ты думаешь, почему я до сих пор не допился до белой горячки?

Я понятия не имел. У каждого это по-разному... А он словно услышал мои мысли.

- Каждый, безусловно, спивается по-разному; я - не такой, как все. Я НИКОГДА не сопьюсь.

»

Я услышал это "никогда" и подумал, что сейчас будет что-то нудное на тему выдающейся индивидуальности и суперорганизма, но его комментарий добил меня окончательно.

- Дело в том, что я очень нужен ТАКОЙ. На крючке.

- Кому? - пожал я плечами.

- Хозяину базы. Точнее, хозяйке.

- И как же она контролирует твое бытовое пьянство?

- Прокрути таблицу, найди строку номер семьдесят шесть...

Я так и сделал; и я знал, что там найду. "Ткачев Михаил Станиславович..." И галочка стояла.

- Что это значит? - спросил я, с прищуром глядя в экран. - Там в строке много непонятного - как и во всей таблице...

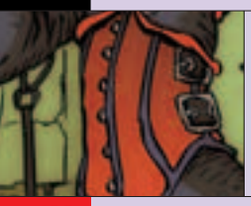
- А это значит, что на мне порча. Меня сделали алкоголиком - и никто не может это изменить...

- Так... - протянул я и отъехал в кресле от стола. Очень интересно. Порча. Средневековье какое-то. Джордано Бруно и инквизиция. Ведьмы и прочая нечисть.

- Не веришь? - ухмыльнулся Ткачев, и я понял, что сейчас разговариваю с абсолютно трезвым человеком. - Понимаешь, она боится компьютера. Но кто-то напел ей, что в наш век высоких технологий даже такие профессии, как ведьма, нуждаются в информационной поддержке. Она пришла и попросила меня написать базу данных. Что-то не очень наворочennое, да ты и сам видишь. Пришла пять с половиной лет назад. Я выполнил ее требования. База получилась хорошая, все работало без ошибок.

Я верил Ткачеву. У него не могло получиться плохо. Вот только странная у заказчицы профессия.

- Конечно же, она не сказала, кто она и зачем в базе нужны какие-то странные значки, немного латыни и пометки на каком-то древнем языке, поддержку которого я сдул из интернета с сайта любителей всей этой оккультной науки.



Я прокрутил таблицу, нашел эти значки, напоминающие клинопись фараонов.

Я прокрутил таблицу, нашел эти значки, напоминающие клинопись фараонов.

- Вижу, - буркнул в трубку и продолжил слушать.

- Ей, наверное, лет пятьдесят или чуть больше. Вроде бы, в таком возрасте не боятся компьютеров панически, как это казалось судя по ее глазам. Она не хотела работать сама, просила меня, обещала платить за ведение всего этого хозяйства. То есть собиралась приходить раз в одну-две недели, приносить мне информацию и ждать, когда же я внесу ее в формы. Поначалу я согласился... А потом, после двух месяцев такого сотрудничества, когда список перевалил за пятьдесят, я нашел в Сети переводы слов, вбитых в базу на латыни. Конечно, перевод приблизительный, поскольку тонкости этого мертвого языка утрачены, но можно перевести так - "сглаз", "порча", "приворот" и еще несколько таких терминов. Я перевел поближе к славянским корням, вполне возможно, что много лет назад все это называлось иначе. Но смысл остается одним и тем же.

Я понял, что слушаю затаив дыхание. Казалось, что я даже перестал моргать.

- Как ты думаешь, что я сделал?

- Ты спросил.

- Точно. И появилась строчка номер семьдесят шесть. Когда она пришла с очередным обновлением, я уже был готов. Готов на все сто два процента. Меня закружило в водовороте клубов, баб, бутылок, стаканов, рюмок и всей этой алкогольной гадости. Когда кончились деньги, кончились и клубы, остался только стакан. И я своими собственными руками внес себя в базу, а она смотрела на меня и стряхивала пепел от своего "Парламента" прямо на пол моей квартиры. Уходя, она сказала: "Пей, не бойся. Не сопьешься". Не оставалось ничего, кроме как поверить...

Я понимал, что тоже начинаю верить, вот только пока еще речь не зашла о том, что ждет такого придурка, как я, который собственными руками внес себя в эту жуткую базу.

- Зачем она все это классифицировала? - спросил я. Вполне разумный вопрос, как мне показалось.

- У нее поразительная работоспособность, - ответил Ткачев. - Такую уйму людей просто невозможно держать ни в голове, ни в записной книжке. Ведь уже скоро закончится третья тысяча... Ты пойми, я ведь пью, но вижу, что происходит на моем компе. Ты сдул все базы...

Я пристыжено смолчал.

- Ладно, молодец. Чего там, учись. Но как ты оказался в этой ведьминой таблице?

- А откуда ты знаешь? - вырвалось у меня. Логично утверждать, что Мишка просто не мог быть в курсе того, что эта строчка появилась.

- Я пробовал это стирать - оно не стирается. Я пробовал изменять - но она очень часто и как-то хитро меняет пароли. Я пробовал уничтожить физически - винчестер оказывался целым...

- А утопить? - спросил я.

- Тетку?

- Винчестер.

- Я с ним даже из комнаты выйти не могу; так что не то чтобы в озере - в ванной не получается.

- Да... - протянул я и вдруг понял, что совершенно свободно разговариваю с Ткачевым на темы, не существовавшие для меня еще полчаса назад. Разговариваю так, будто мы с ним в кино собрались или в театр и думаем, кого бы из девчонок пригласить. - Так откуда ты знаешь, что я там есть?

- Вот тут самое главное. Я пробовал копировать - это единственное, что получается. И лучше бы я этого не делал - потому что эти базы начинают жить самостоятельной жизнью.

- То есть? - я понимал все меньше и меньше, но интересу наравне со страхом во мне появлялось все больше и больше.

- То есть изменения, сделанные в одной, появлялись везде, в каждой таблице. Примерно год назад она все-таки купила себе компьютер. Я скопировал, показал, объяснил. И попросил у нее снять с меня это дерьмо. Она только рассмеялась.

- И она оставила копию базы тебе? - не понял я Ткачева.

- Да.

- Зачем?

- Чтобы я никогда не бросил пить - даже если она порчу снимет.

- В смысле?

- Я ведь вижу, что она делает. Просто я не могу предупредить этих людей - они появляются в базе постфактум, когда работа уже сделана. Она уже соорудила очередной сглаз, приворот или еще какую пакость, поставила галочку и спокойно пошла спать. Я смотрю, как еще один человек загибается, а сделать ничего не могу...

- Я... Понимаешь, я сам... Сам себя впечатал. Так, шутки ради. Я же не знал...

- Сам?! - Ткачев чуть не задохнулся на том конце линии. - Ты... Придурок! Идиот! Да ты...

У него не было слов. У меня тоже. И я вдруг понял, что мы оба - я здесь и он там - мы оба смотрим на пустой чек-бокс для галочки и понимаем, что где-то далеко, в ведьминой квартире, в базе возникла еще одна строка, которая ждет своего часа.

- Миша, кто она? - спросил я. - Я думаю, что можно что-то сделать. Ее надо найти, ее надо остановить...

- "Ночного Дозора" начитался? - ехидно спросил Ткачев. - Ее никто не остановит.

- Но ты же знаешь, кто она и где живет, так скажи. Я пойду сам...

- А вот насчет "скажи" у нее тут целая система разработана, - вздохнул Мишка.

- Какая?

- А чтоб сказать не смог.

- Не сможешь?

- Смог бы - сам бы давно убил. Черт побери, мне так выпить хочется... У тебя пива не осталось?

- Нет, - ответил я и понял, что он не скажет...

Через секунду в трубке раздались гудки. Что ж, он и так много сделал. По крайней мере, он не смолчал - он позвонил. Представляю, как его там сейчас заворачивает в дугу. Теперь запьет на неделю, а ведь сказал всего лишь маленькую часть правды...

С тех пор прошло три дня. Я решился выйти из дому только вчера. Почему-то вспомнил слова Ткачева и посмотрел "Ночной Дозор". А потом вернулся и не отрываясь смотрел на свою строчку в таблице. Галочки не было.

Ее нет и сейчас, когда я пишу эти строки. Никто не знает, когда ведьма обратит внимание на то, что в таблице появился кто-то без ее ведома. И неизвестно еще, чем она наградит МЕНЯ.

А у Мишки все время "занято"...

LG
Life's Good

FLATRON™
freedom of mind

FLATRON F700P

Абсолютно плоский экран
Размер точки 0,24 мм
Частота развертки 95 кГц
Экранное разрешение 1600x1200
USB-интерфейс

Dina Victoria
(095) 688-61-17, 688-27-65
WWW.DVCOMP.RU

Москва: АБ-групп (095) 745-5175; Акситек (095) 784-7224; Банкос (095) 128-9022; ДЕЛ (095) 250-5536; Дилайн (095) 969-2222; Инкотрейд (095) 176-2873; ИНЭЛ (095) 742-6436; Карин (095) 956-1158; Компьютерный салон SMS (095) 956-1225; Компания КИТ (095) 777-6655; Никс (095) 974-3333; ОЛДИ (095) 105-0700; Регард (095) 912-4224; Сетевая Лаборатория (095) 784-6490; СКИД (095) 232-3324; Тринити Электроникс (095) 737-8046; Формоза (095) 234-2164; Ф-Центр (095) 472-6104; ЭЛСТ (095) 728-4060; Flake (095) 236-992; Force Computers (095) 775-6655; ISM (095) 718-4020; Meijin (095) 727-1222; NT Computer (095) 970-1930; R-Style Trading (095) 514-1414; USN Computers (095) 755-8202; ULTRA Computers (095) 729-5255; ЭЛЕКТОН (095) 956-3819; ПортКом (095)777-0210; **Архангельск:** Северная Корона (8182) 653-525; **Волгоград:** Техком (8612) 699-850; **Воронеж:** Рет (0732) 779-339; РИАН (0732) 512-412; Сани (0732) 54-00-00; **Иркутск:** Билайн (3952) 240-024; Комтек (3952) 258-338; **Краснодар:** Игрек (8612) 699-850; **Лабытнанги:** КЦ ЯМАЛ (34992) 51777; **Липецк:** Регард-тур (0742) 485-285; **Новосибирск:** Квеста (38322) 332-407; **Нижний Новгород:** Бюро-К (8312) 422-367; **Пермь:** Гаском (8612) 699-850; **Ростов-на-Дону:** Зенит-Компьютер (8632) 950-300; **Тюмень:** ИНЭКС-Техника (3452) 390-036.

ЧИТАЙТЕ В ФЕВРАЛЕ:

«Ночной дозор»
- Только в «РС ИГРАХ». Эксклюзивная информация о новом проекте Nival Interactive: обзор текущей версии игры, видеорепортаж, интервью с командой и дневники разработчиков, конкурс.

Nexus: The Jupiter Incident
- Игра месяца! Лучшая космическая стратегия!

Chronicles of Riddick: Escape from Butchers Bay
- Первый кандидат на звание «Блокбастер года»!

РС ИГРЫ

ПРАВИЛЬНЫЙ ЖУРНАЛ О КОМПЬЮТЕРНЫХ ИГРАХ

Правильная комплектация
Двухслойный DVD или 3 CD

Правильный объем
240 страниц

ФЕВРАЛЬСКИЙ НОМЕР УЖЕ В ПРОДАЖЕ

ЧАСТЬ ТИРАЖА – С DVD
8.5Gb
ЭКСКЛЮЗИВНОЕ ВИДЕО!!!

А ТАКЖЕ:
- Дневники разработчиков. Куда исчезли «Корсары 2»?
- Спец-тема. Оружие, которое нас впечатлило!
- Разговор по душам. Американ МакГи - благопристойный хулиган.
- Рецензии на Prince of Persia: Warrior Within, LOTR: Battle for Middle-Earth, Pro Evolution Soccer 4, Sid Meier's Pirates, EverQuest 2...

И многое другое!

Никакого мусора и невнятных тем, настоящий геймерский рай
ТОЛЬКО РС ИГРЫ

ЕСЛИ ТЫ ГЕЙМЕР - ТЫ НЕ ПРОПУСТИШЬ!

(game)land