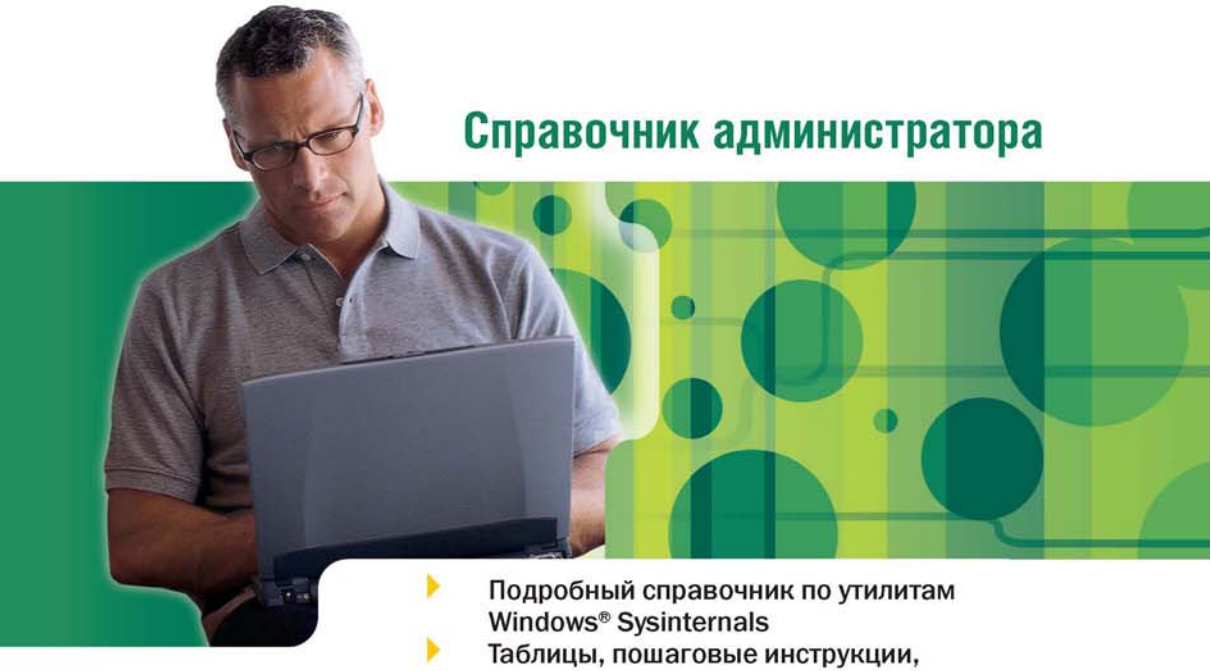
Марк Руссинович
Аарон Маргозис
Предисловие Дэвида Соломона

Утилиты Sysinternals

Справочник администратора



- Подробный справочник по утилитам Windows® Sysinternals
- Таблицы, пошаговые инструкции, описание параметров

IT Professional

РУССКАЯ РЕДАКЦИЯ Microsoft bhv

Моим коллегам —
специалистам по устранению неполадок Windows.
Никогда не отступайте и не сдавайтесь!
– Марк Руссинович

Элизе, благодаря ей сбываются самые прекрасные мечты!
(И она гораздо круче меня!)
– Аарон Маргозис

Mark Russinovich
Aaron Margosis

Windows® Sysinternals Administrator's Reference

Microsoft® Press

Марк Руссинович
Аарон Маргозис
Предисловие Дэвида Соломона

Утилиты Sysinternals

Справочник администратора

РУССКАЯ РЕДАКЦИЯ bhv®

2012

УДК 004.738.5
ББК 32.973.202
Р89

Руссинович Марк, Маргозис Аарон

Р89 Утилиты Sysinternals. Справочник администратора. / Пер. с англ. — М. : Издательство «Русская редакция» ; СПб. : БХВ-Петербург, 2012. — 480 стр. : ил.

ISBN 978-5-7502-0411-3 («Русская редакция»)
ISBN 978-5-9775-0826-1 («БХВ-Петербург»)

Эта книга — исчерпывающее руководство по использованию утилит Sysinternals. Авторы книги — создатель утилит Sysinternals Марк Руссинович и признанный эксперт по Windows Аарон Маргозис — подробно разбирают многочисленные функции утилит для диагностики и управления файлами, дисками, системой безопасности и встроенным инструментарием Windows. Рекомендации авторов проиллюстрированы многочисленными примерами из реальной жизни. Изучив их, вы сможете справиться с неполадками в ИТ-системах так, как это делают настоящие профессионалы.

Книга состоит из 18 глав и предметного указателя. Она предназначена для ИТ-специалистов и опытных пользователей Windows, которые хотят применять утилиты Sysinternals с максимальной эффективностью.

УДК 004.738.5
ББК 32.973.202

РУССКАЯ РЕДАКЦИЯ bhv®

Оглавление

Предисловие

Просьба Марка и Аарона написать вступление для этой книги — большая честь для меня. Я познакомился с Марком и его утилитами в 1997 году, когда впервые услышал его выступление на конференции разработчиков Windows в Санта Клара, Калифорния. Тогда я не знал, что через два года мы будем вместе работать над «Inside Windows 2000» и следующими редакциями «Windows Internals».

Из-за того что я работал с Марком над книгами, а потом над курсами по Windows Internals, которые мы вместе создавали и вели, дифирамбы за утилиты Sysinternals часто пели мне (что не очень нравилось Марку). Конечно, благодарности хотелось принять, но правда такова: я постоянно пользуюсь этими утилитами, но не создал ни одну из них.

Потребность в книге по утилитам Sysinternals назрела уже очень давно, но факт, что их успешно используют и *без* книги, объясняющей принципы их работы, говорит сам за себя. Книга еще шире раскрывает перед ИТ-специалистами возможности утилит Sysinternals, позволяет заглянуть внутрь Windows и по-настоящему понять, что там происходит. Аарон Маргозис подверг эти утилиты дотошному анализу, что позволило внести многие улучшения, исправить неточности, дополнить справочный текст и добавить новые функции.

При помощи этих утилит я выловил и исправил огромное количество ошибок в системах и приложениях, даже в отчаянных ситуациях, когда я просто не думал, что от этих утилит будет толк. В результате я пришел к такому выводу: если сомневаешься, запускай Filemon и Regmon (вместо них теперь Procmon).

Чтобы научить читателей применять утилиты для решения реальных проблем, в книге есть целый раздел с «рецептами» решения типичных задач. Эти примеры из жизни помогут ИТ-специалистам в работе с утилитами Sysinternals при устранении неполадок, кажущихся неустранимыми.

Под занавес небольшое предупреждение: не помню, сколько раз я писал Марку об ошибках (которые он уже исправил). Поэтому, прежде чем отправлять ему письмо об очередном баге, убедитесь, что используете последние версии утилит. Для этого лучше всего следить за RSS-каналом сайта Sysinternals. Словом, эта книга должна быть настольной у каждого ИТ-специалиста. А если увидите Марка, непременно скажите ему, как высоко вы цените работу Дэйва над утилитами Sysinternals!

Дэвид Соломон, Президент компании David Solomon
Expert Seminars, Inc., www.solsem.com

Введение

Пакет Sysinternals включает более 70 утилит для диагностики и устранения неполадок платформы Windows, написанных мной (Марком Руссиновичем) и Брайсом Когсвеллом. С 2006 года (после приобретения корпорацией Microsoft фирмы Sysinternals) эти утилиты доступны для бесплатной загрузки на сайте Microsoft Windows Sysinternals (в составе Microsoft TechNet).

Цель этой книги — познакомить вас с утилитами Systinternals и помочь понять, как использовать их максимально эффективно. Также мы покажем, как с помощью этих утилит устранять неполадки Windows.

Книга написана в соавторстве с Аароном Магозисом, но повествование ведется от моего лица. Однако это нисколько не умаляет вклад Аарона, без его кропотливой работы книга не увидела бы свет.

Утилиты

В этой книге описываются все утилиты Sysinternals, доступные на вебсайте Windows Sysinternals (*http://technet.microsoft.com/en-us/sysinternals/default.aspx*) и все их функции, поддерживавшиеся на момент написания книги (лето 2011). Однако пакет Sysinternals очень динамично развивается: в постоянно добавляются новые функции, а время от времени появляются и новые утилиты (следите за RSS-каналом блога «Sysinternals Site Discussion»: *http://blogs.technet.eom/b/sysinternals*). Поэтому, к моменту, когда вы начнете читать эту книгу, кое-какие сведения могут уже устареть. Следите за обновлениями, чтобы не пропустить новые функции и исправления ошибок.

В книге не описываются устаревшие утилиты, исключенные из пакета Sysinternals и списка на официальном сайте. Если вы все еще используете RegMon (Registry Monitor) или FileMon (File Monitor), замените их утилитой Process Monitor, описанной в главе 4. Rootkit Revealer, одна из первых утилит для обнаружения руткитов (именно с ее помощью был обнаружен известный руткит Sony), исчерпала себя и отправлена «на пенсию». Некоторые другие утилиты (такие как Newsid и EfsDump), генерировавшие уникальные значения, исключены за ненадобностью (соответствующие функции теперь встроены в Windows).

История Sysinternals

Первая утилита Sysinternals, которую я написал, Ctrl2cap, родилась из необходимости. Прежде чем перейти на Windows NT в 1995 г., я работал, в основном, с UNIX-системами. У них на клавиатуре клавиша Ctrl находилась там, где на стандартной клавиатуре PC находится Caps Lock. Я не стал привыкать к новой раскладке, а разобрался с драйверами устройств Windows NT и написал драйвер, преобразующий нажатие Caps Lock в нажатие Ctrl. Утилита Ctrl2cap до сих пор есть на сайте Sysinternals, а я до сих пор использую ее на всех своих компьютерах.

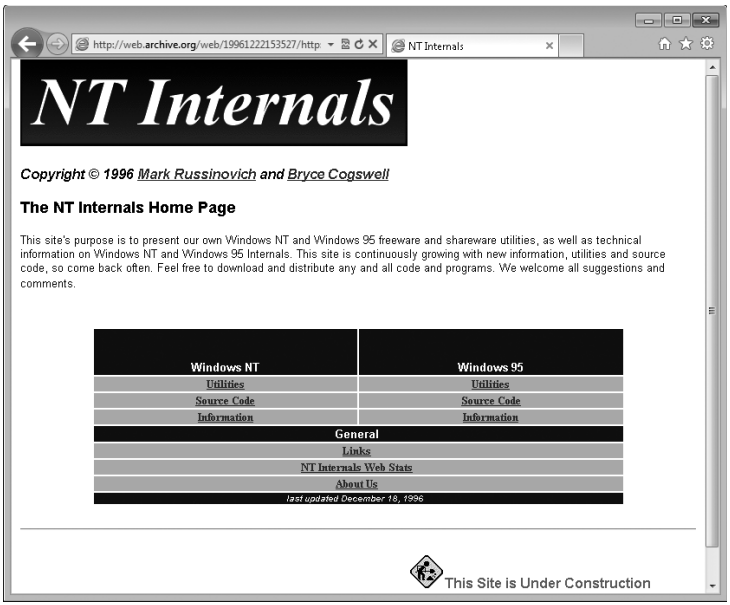
Это была первая из многих утилит, написанных мной, чтобы разобраться во внутреннем устройстве Windows NT и попутно сделать что-то полезное. Следующую утилиту, NTFSDOS, я создал вместе с Брайсом Когсвеллом. Я познакомился с Брайсом в магистратуре Университета Карнеги-Меллона, где мы вместе работали и запустили проект по разработке ПО для Windows 3.1. У меня возникла идея утилиты, позволяющей читать данные с разделов NTFS после загрузки с обычной дискеты DOS. Брайс решил, что это интересная задача для программиста, мы разделили между собой работу и примерно через месяц выпустили первую версию.

Следующие две утилиты, Filemon и Regmon, я также написал вместе с Брайсом. Утилиты NTFSDOS, Filemon и Regmon стали основой Sysinternals. Утилиты Filemon и Regmon, созданные для Windows 95 и Windows NT, показывали активность файловой системы и реестра. Это были первые утилиты подобного рода, что сделало их незаменимыми инструментами для решения разного рода проблем.

Мы с Брайсом решили сделать наши утилиты общедоступными, но у нас не было собственного веб-сайта, поэтому сначала мы опубликовали их на сайте друга, Эндрю Шульмана, с которым я познакомился на почве его собственных исследований внутреннего устройства DOS и Windows 95. Такое посредничество не позволяло нам обновлять утилиты и исправлять ошибки так быстро, как мы того хотели, поэтому в сентябре 1996 г. мы создали свой сайт, NTInternals.com, на котором размещали утилиты и статьи о внутреннем устройстве Windows 95 и NT. Также мы разработали несколько утилит, которые хотели продавать за деньги, поэтому тогда же мы основали компанию Winternals Software, которую рекламировали баннером на NTInternals.com. Первой утилитой, вышедшей под «крышей» Winternals Software была NTRecover, позволяющая подключать диски систем Windows NT от переставших загружаться компьютеров к исправным и работать с ними как обычно.

Миссией NTInternals.com было распространение бесплатных утилит, в которых наше глубокое понимание устройства ОС Windows воплощалось в средства для диагностики, мониторинга и управления. За несколько месяцев посещаемость сайта достигла 1500 пользователей в день, и он стал одним из самых популярных сайтов с утилитами для Windows (см. рис.). В 1998 г. по «совету» юристов Microsoft нам пришлось сменить название на *Sysinternals.com*.

В последующие несколько лет утилиты продолжали развиваться. Мы добавляли новые утилиты, часто в ответ на предложения пользователей или создания нового способа предоставления информации о «внутренностях» Windows.



Сайт NT Internals, как он выглядел в декабре 1996 г. (Из Internet Archive: Wayback Machine)

Утилиты Sysinternals разделились на три основные категории: для программистов, для решения проблем и для управления системами. Утилита DebugView, перехватывающая и показывающая информацию об отладке программ, была одним из ранних инструментов для программистов. Я написал ее для себя, занимаясь разработкой драйверов устройств. Первые утилиты для устранения неполадок — DLLView, отображающая библиотеки DLL, загруженные процессами, и HandleEx, GUI-утилита, показывающая открытые описатели (В 2001 г. я объединил их функции в Process Explorer.) Утилиты PsTools (см. главу 6) — одни из наиболее популярных утилит для управления, собранные в единый пакет для удобства. PsList, первая версия PsTool, напоминала команду «ps», которая в UNIX показывает список процессов. Число утилит росло, появлялись новые функции, а в итоге получился пакет, позволяющий легко решать многие задачи на удаленных системах, не устанавливая на них специальное ПО.

В 1996 г. я также начал писать для журнала «Windows IT Pro», рассказывая о внутреннем устройстве Windows и утилитах Sysinternals. Одна из моих статей вызвала бурное обсуждение и привлекла ко мне внимание Microsoft (впрочем, не факт, что благосклонное). В этой статье, вышедшей под заголовком «Inside the Difference Between Windows NT Workstation and Windows NT Server» («Внутренние различия Windows NT Workstation и Windows NT

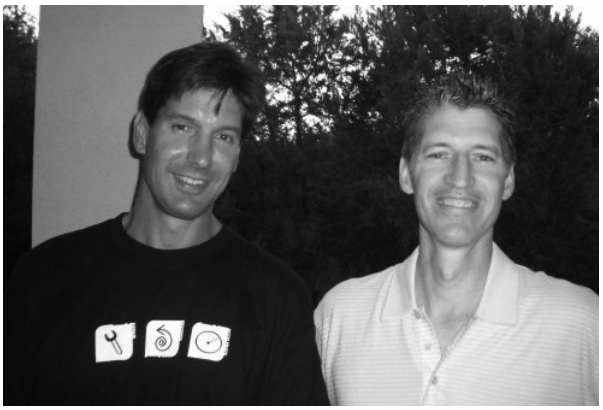
Server»), говорится о минимальных различиях межу NT Workstation и NT Server, что противоречило маркетинговой политике Microsoft.

По мере развития утилит я начал думать о написании книги о внутреннем устройстве Windows. Такая книга уже существовала: «Inside Windows NT» (Microsoft Press, 1992). Первое издание, написанное Хелен Кастер, было приурочено к выпуску Windows NT 3.1. Второе издание было переписано с учетом изменений, появившихся в Windows NT 4.0, Дэвидом Соломоном — признанным экспертом по операционным системам, преподавателем и писателем, с которым я работал в DEC. Чтобы не писать новую книгу в одиночку, я предложил Дэвиду вместе работать над третьим изданием, посвященным Windows 2000. Мои отношения с Microsoft понемногу налаживались после той статьи, благодаря сообщениям о багах Windows, которые я отправлял напрямую разработчикам, но Дэвиду все равно пришлось получать «допуск», который ему дали.

В результате мы с Дэвидом Соломоном написали третье, четвертое и пятое издания. Начиная с четвертого, книга получила название «Windows Internals» (пятое издание опубликовано в 2009 г.). Вскоре после окончания работы над «Inside Windows 2000» (Microsoft Press, 2000) я присоединился к Дэвиду в преподавании на семинарах по внутреннему устройству Windows, добавив собственные темы. Эти семинары велись по всему миру и даже в офисах Microsoft для разработчиков Windows. На них широко использовались утилиты Sysinternals, позволяющие заглянуть глубоко внутрь Windows и узнать много нового. Дэвид до сих пор ведет семинары, о которых можно узнать на сайте *http://www.solsem.com*.

К 2006 г. мои отношения с Microsoft полностью наладились, моя компания Winternals предлагала полную линейку приложений для управления инфраструктурой предприятия, ее штат вырос до 100 сотрудников, а утилиты Sysinternals достигли двух миллионов скачиваний в месяц. 18 июля 2006 г. Microsoft выкупила Winternals и Sysinternals. Вскоре после этого мы с Брайсом переехали в Редмонд и влились в команду Windows. Сегодня я работаю одним из технических сотрудников (Technical Fellows) Microsoft (которых в компании совсем немного), помогая направлять развитие компании в техническом плане. Сейчас я — в группе Windows Azure и работаю над ядром операционной системы Microsoft для облачных вычислений.

Одной из целей продажи Microsoft фирмы Sysinternals была гарантия дальнейшей бесплатной доступности наших утилит и развития созданного нами сообщества. И эта цель достигнута. Сегодня сайт Windows Sysinternals на technet.microsoft.com является одним из наиболее часто посещаемых: 50000 посетителей в день и три миллиона скачиваний в месяц. Опытные пользователи Sysinternals периодически заходят за последними версиями и новыми утилитами, такими как RAMMap и VMMap, а также для участия в жизни сообщества Sysinternals и общения на растущем форуме (30000 зарегистрированных пользователей на момент написания книги). Я продолжаю улучшать существующие утилиты и добавлять новые, включая инструменты для работы в Windows Azure.



Марк Руссинович и Брайс Когсвелл, 2006 г.

Многие (в том числе и Дэвид Соломон), говорили, что книга по этим утилитам была бы полезной, и ее давно пора написать. Моя работа в Microsoft не оставляла времени для этого, и Дэвид посоветовал найти помощника. Я рад, что Аарон Маргозис согласился на это. Аарон — ведущий консультант служб взаимодействия с клиентами Microsoft Public Sector, известный своим глубоким знанием системы безопасности Windows и вопросов совместимости приложений. Я знаю Аарона много лет, а его писательский талант, знание внутреннего устройства Windows и опыт работы с утилитами Sysinternals сделали его идеальным соавтором.

Кому адресована эта книга

Эта книга написана для ИТ-специалистов и опытных пользователей Windows, которые хотят применять утилиты Sysinternals с максимальной эффективностью. Независимо от вашего опыта работы с этими утилитами и систем, которыми вы управляете, вы обнаружите новые возможности, получите советы и узнаете о приемах, которые позволят эффективнее решать сложнейшие проблемы с Windows и упростить управление и мониторинг систем.

Требования к читателям

Предполагается, что вы знакомы с операционными системами Windows. Желательно владеть фундаментальными понятиями, такими как «процесс», «поток», «виртуальная память» и командная строка Windows, некоторые из них обсуждаются в главе 2.

Структура книги

Книга разделена на три части. В первой части приводится обзор утилит Sysinternals и веб-сайта Sysinternals, описываются общие функции всех утилит, рассказывается, куда обращаться за помощью, и приводится общая

информация об устройстве Windows, помогающая лучше понять принципы работы этой платформы и сведения, предоставляемые утилитами.

Вторая часть содержит подробную справочную информацию по всем функциям утилит Sysinternals, параметрам командной строки, требованиям и прочим аспектам. Эта часть снабжена массой иллюстраций и примеров, она даст ответ почти на любой вопрос об утилитах. Основным утилитам, таким как Process Explorer и Process Monitor, посвящены отдельные главы, остальные утилиты описываются по категориям, как, например, утилиты для обеспечения безопасности, утилиты для работы с Active Directory и утилиты для работы с файлами.

В третьей части описаны реальные случаи устранения неполадок с использованием утилит Sysinternals, из моего с Аароном опыта, а также присланные администраторами и пользователями из разных уголков мира.

Принятые обозначения

Следующие элементы оформления информации призваны сделать текст более понятным и удобным для чтения:

- врезки «Примечание» содержат дополнительную информацию и альтернативные способы решения задач;
- текст, вводимый с клавиатуры (кроме фрагментов кода), выделяется полужирным шрифтом;
- знак плюс (+) между названиями клавиш означает, что эти клавиши нужно нажать одновременно. Например, «нажмите Alt+Tab» означает, что вы должны нажать и удерживать клавишу Alt, а затем нажать клавишу Tab;
- вертикальная черта между элементами меню (например, Файл | Закрыть) оз-начает, что следует выбрать первый элемент меню, затем следующий и т. д.

Требования к системе

Утилиты Sysinternals работают в следующих версиях Windows, включая 64-разрядные, если не сказано обратное:

- Windows XP с третьим пакетом обновлений;
- Windows Vista;
- Windows 7;
- Windows Server 2003 со вторым пакетом обновлений;
- Windows Server 2003 R2;
- Windows Server 2008;
- Windows Server 2008 R2.

Для запуска некоторых утилит требуются права администратора, отдельные функции некоторых утилит также требуют прав администратора.

Благодарности

В первую очередь, мы хотим поблагодарить Брайса Когсвелла, сооснователя Sysinternals, за его огромный вклад в развитие утилит Sysinternals. Благодаря нашему замечательному сотрудничеству, содержимое сайта Sysinternals — это больше, нежели просто сумма наших индивидуальных усилий. Брайс ушел из Microsoft в октябре 2010 г., и мы желаем ему удачи во всех его начинаниях.

Мы благодарим Дэвида Соломона за то, что он подтолкнул Марка к написанию этой книги, предоставил подробные рецензии многих глав и написал вступление. Кроме того, Дэйв долгие годы был одним из самых эффективных пропагандистов Sysinternals и предложил много полезных функций.

Спасибо Кертису Метцу и Карлу Зенгу, которые управляют веб-сайтом Sysinternals, форумами и публикацией кода. Когда Microsoft приобрела Sysinternals, эту роль выполнял Отто Хелвег, и мы благодарим его за то, что он помог сохранить дух Sysinternals во время слияния.

Благодарим людей, предоставивших ценные и подробные рецензии, исправления и предложения по книге: это Андреас Кляйн, Брайан Матюш, Бруно Алейзо, Карстен Киндер, Крис Джексон, Эван МакКеллар, Фатих Колгар, Гаутам Ананд, Говри Кумар Чандрамули, Грег Коттингем, Джон Дитрик, Марио Хевардт, Марио Ракканьи, Марк Прим, Мэтт Гарсон, Павел Лебединский, Ричард Дайвер, Скотт Фрунци, Стивен Гриффин, Тим Рекмайер. Эндрю Ричардс заслуживает отдельного упоминания за то, что отрецензировал больше глав, чем все остальные.

Спасибо Карлу Харрисону за материалт об использовании LiveKd для получения дампов ядра на работающей системе.

Благодарим Мартина ДелРе из Microsoft Press за то, что увидел потенциал этой книги; Девона Масгрейва, тоже из Microsoft Press, за поддержку книги; и Стива Сэгмана из Waypoint Press за сопровождение книги при редактировании и издании. Спасибо Кристофу Назарре за техническую правку и Роджеру ЛеБланку за правку текста.

Элиза, жена Аарона, заслуживает благодарности за поддержку Аарона в критические моменты работы над книгой. Аарон благодарит ее и детей за поддержку, а Бренду Шрайер — за фотографию автора.

Марк благодарит жену Дэрил и дочь Марию за поддержку всех его начинаний.

Список ошибок и поддержка книги

Мы приложили все усилия, чтобы обеспечить точность содержимого книги и приложений. Ошибки, обнаруженные после издания книги на английском языке, будут перечислены на сайте Microsoft Press:

http://go.microsoftcom/FWLink/?Linkid=220275

Если вам требуется дополнительная помощь, напишите в службу поддержки Microsoft Press по адресу *msinput@microsoft.com* (обратите внимание: по указанным адресам поддержка приложений Microsoft не предоставляется). Следите также за Microsoft Press в Твиттере: *http://twitter.com/MicrosoftPress.*

Часть I

Приступая к работе

Глава 1

Введение в утилиты Sysinternals

Утилиты Sysinternals — это бесплатные усовершенствованные программы для управления, диагностики и устранения неполадок компьютеров на платформе Microsoft Windows. Их написали основатели компании Sysinternals: я — Марк Руссинович, и Брюс Когсвелл[1]. После выкупа фирмы Sysinternals корпорацией Microsoft в июле 2006 года эти утилиты доступны для скачивания с веб-сайта TechNet.

Утилиты Sysinternals:

- предоставляют ИТ-специалистам и разработчикам многие функции, давно ожидаемые ими;
- интуитивно-понятны и просты в использовании;
- созданы в виде автономных исполняемых образов, не требующих установки. Их можно запускать откуда угодно, в том числе из сетевой папки или со съемного носителя;
- после завершения не оставляют в системе «мусорных» данных и прочих следов.

Поскольку в Sysinternals нет большой команды разработчиков, я оперативно выпускаю новые компоненты, утилиты и вношу исправления. В некоторых случаях я могу менее чем за неделю создать полезный и простой в использовании компонент.

С другой стороны, по той же самой причине — из-за отсутствия большого штата тестировщиков и полноценной процедуры тестирования — утилиты предлагаются «как есть», без официальной поддержки продукции Microsoft. Тем не менее, команда Sysinternals старается оказывать пользователям поддержку через специализированный форум на сайте компании (о нем см. ниже), и я стараюсь, по возможности, быстрее исправлять обнаруженные ошибки.

---

[1] Брюс ушел из Microsoft в конце 2010 года и больше не принимает участие в разработке утилит Sysinternals.

Обзор утилит

Утилиты Sysinternals охватывают многие аспекты ОС Windows. Одни утилиты обладают весьма широким функционалом, например Process Explorer и Process Monitor, попадающим в разные категории, а другие довольно специализированы, их можно отнести к категории анализаторов процессов или файловых утилит. Многие утилиты оснащены графическим интерфейсом пользователя (GUI), другие являются консольными программами с интерфейсом командной строки и предназначены для использования в сценариях или ручного запуска в командной строке.

Эта книга начинается с описания трех основных утилит (Process Explorer, Process Monitor и Autoruns), каждой из них посвящена отдельная глава. В следующих главах рассматриваются категории, каждая из которых включает несколько логически связанных утилит.

В табл. 1-1 приводится перечень глав с кратким обзором рассматриваемых в них утилит.

Табл. 1-1. Темы глав книги

Утилита	Описание
Глава 3. Process Explorer	
Process Explorer	Заменяет Диспетчер задач, отображая более подробную информацию о процессах и потоках, включая их родство, загруженные DLL и открытые описатели объектов, таких как файлы
Глава 4. Process Monitor	
Process Monitor	Регистрирует активность файловой системы, реестра, сети, процессов, потоков, а также загрузку образов в реальном времени
Глава 5. Autoruns	
Autoruns	Перечисляет и определяет программное обеспечение, настроенное на автоматический запуск при загрузке системы, входе в нее пользователя и запуске Internet Explorer, а также позволяет отключать и удалять эти элементы
Глава 6. PsTools	
PsExec	Запускает процессы удаленно и (или) в локальной системе, перенаправляя их вывод
PsFile	Перечисляет и закрывает удаленно открытые файлы
PsGetSid	Отображает идентификатор защиты (SID) участника системы безопасности (компьютера, пользователя, группы или службы)
Pslnfo	Выводит информацию о системе
PsKill	Завершает процессы, заданные по имени или идентификатору (PID)
PsList	Выводит подробную информацию о процессах и потоках
PsLoggedOn	Перечисляет учетные записи пользователей, выполнивших вход в систему по локальной сети и через удаленные подключения

Табл. 1-1. (*продолжение*)

Утилита	Описание
PsLogList	Выводит записи журнала событий
PsPasswd	Изменяет пароли учетных записей пользователей
PsService	Перечисляет службы Windows и позволяет управлять ими
PsShutdown	Завершает работу, выполняет выход и изменяет режим энергопотребления локальных и удаленных систем
PsSuspend	Приостанавливает и возобновляет процессы
Глава 7. Утилиты для работы с процессами и диагностики	
VMMap	Отображает сведения об использовании виртуальной и физической памяти процессом
ProcDump	Создает дамп памяти процесса в заданной ситуации, например при пиковой загруженности ЦП или «зависании» окон
DebugView	Отслеживает вывод отладчика режимов пользователя и ядра на локальном или удаленном компьютере
LiveKd	Запускает стандартный отладчик ядра на «снимке» работающей локальной системы или образа в Hyper-V без перезагрузки в режиме отладки и записывает дамп памяти функционирующей системы
ListDLLs	Отображает в окне консоли информацию о DLL, загруженных системой
Handle	Отображает в консольном окне информацию об описателях объектов, открытых процессами
Глава 8. Утилиты системы безопасности	
SigCheck	Проверяет цифровые подписи и отображает информацию о версии
AccessChk	Ищет объекты, предоставляющие разрешения отдельным пользователям или группам, а также отображает подробную информацию о предоставленных разрешениях
AccessEnum	Ищет файлы и разделы реестра, выявляет места возможного изменения разрешений
ShareEnum	Перечисляет общие файлы и принтеры, используемые совместно в сети, а также пользователей, имеющих к ним доступ
ShellRunAs	Восстанавливает возможность запуска программы под другой учетной записью пользователя в Windows Vista
Autologon	Настраивает учетную запись пользователя для автоматического входа при загрузке системы
LogonSessions	Перечисляет активные сеансы LSA на компьютере
SDelete	Надежно удаляет файлы и папки, стирает данные в нераспределенных областях жесткого диска

Табл. 1-1. (*продолжение*)

Утилита	Описание
Глава 9. Утилиты для работы с Active Directory	
AdExplorer	Отображает и включает редактирование объектов Active Directory
Adlnsight	Отслеживает вызовы LDAP API службы каталогов Active Directory
AdRestore	Перечисляет и восстанавливает удаленные объекты Active Directory
Глава 10. Утилиты рабочего стола	
Bglnfo	Отображает информацию о конфигурации компьютера на обоях рабочего стола
Desktops	Запускает приложения на отдельных виртуальных рабочих столах
Zoomit	Увеличивает размер экрана и включает аннотацию экрана
Глава 11. Утилиты для работы с файлами	
Strings	Ищет в файлах текст ASCII или Unicode
Streams	Находит объекты файловой системы, имеющие альтернативные потоки данных, и удаляет эти потоки
junctions	Перечисляет и удаляет символические ссылки
FindLinks	Перечисляет жесткие ссылки NTFS
DU	Перечисляет логические размеры и размеры на диске папок с содержимым
PendMoves	Сообщает об операциях с файлами, запланированных для выполнения во время следующей загрузки системы
Movefile	Планирует операции с файлами для выполнения во время следующей загрузки системы
Глава 12. Утилиты для работы с диском	
Disk2Vhd	Захватывает VHD-образ физического диска
Diskmon	Регистрирует активность жесткого диска на уровне секторов
Sync	Сбрасывает дисковый кеш на физический диск
DiskView	Отображает графическую карту кластеров тома с указанием принадлежности кластеров файлам
Contig	Дефрагментирует заданные файлы или показывает фрагментацию того или иного файла
PageDefrag	Дефрагментирует во время загрузки системы те системные файлы, которые невозможно дефрагментировать во время работы Windows
DiskExt	Отображает информацию об экстентах диска

Табл. 1-1. (*окончание*)

Утилита	Описание
LDMDump	Отображает подробную информацию о динамических дисках из базы данных Диспетчера логических дисков (LDM)
VolumeID	Изменяет идентификатор (серийный номер) тома
Глава 13. Сетевые и коммуникационные утилиты	
TCPView	Перечисляет активные конечные точки TCP- и UDP-соединений
Whois	Сообщает информацию о регистрации доменов Интернета или выполняет обратный просмотр DNS
Portmon	Отслеживает ввод-вывод через последовательные и параллельные порты в реальном времени
Глава 14. Утилиты для получения информации	
RAMMap	Предоставляет подробную схему использования физической памяти
CoreInfo	Перечисляет сопоставления логических процессов с ядрами, сокетами, узлами доступа к неоднородной памяти (NUMA) и группами процессоров
ProcFeatures	Выводит список функций процессора, например, защиту памяти No-Execute
WinObj	Отображает пространство имен Диспетчера объектов Windows
LoadOrder	Показывает примерный порядок загрузки драйверов устройств и запуска служб Windows
PipeList	Перечисляет именованные каналы, прослушиваемые системой
ClockRes	Показывает текущее, максимальное и минимальное разрешения часов системы
Глава 15. Другие утилиты	
RegJump	Запускает RegEdit и открывает указанный путь реестра
Hex2Dec	Преобразует шестнадцатеричные числа в десятеричные и обратно
RegDelNull	Выполняет поиск и удаление разделов реестра с NULL-именами
Bluescreen Screen Saver	Экранная заставка, убедительно имитирующая «синий экран смерти»
Ctrl2Cap	Преобразует нажатия Caps Lock в нажатия Ctrl

Сайт Windows Sysinternals

Проще всего попасть на сайт Sysinternals (рис. 1-1) — набрать адрес *http://www.sysinternals.com*, откуда вы будете перенаправлены на домашнюю страницу сообщества Sysinternals на Microsoft TechNet, находящуюся в данный момент по адресу *http://technet.microsoft.com/sysinternals*. Помимо полного набора утилит Sysinternals, на сайте есть ссылки на множество тематических

ресурсов: обучающие курсы, книги, блоги, статьи, телеконференции, новости и форум сообщества Sysinternals.

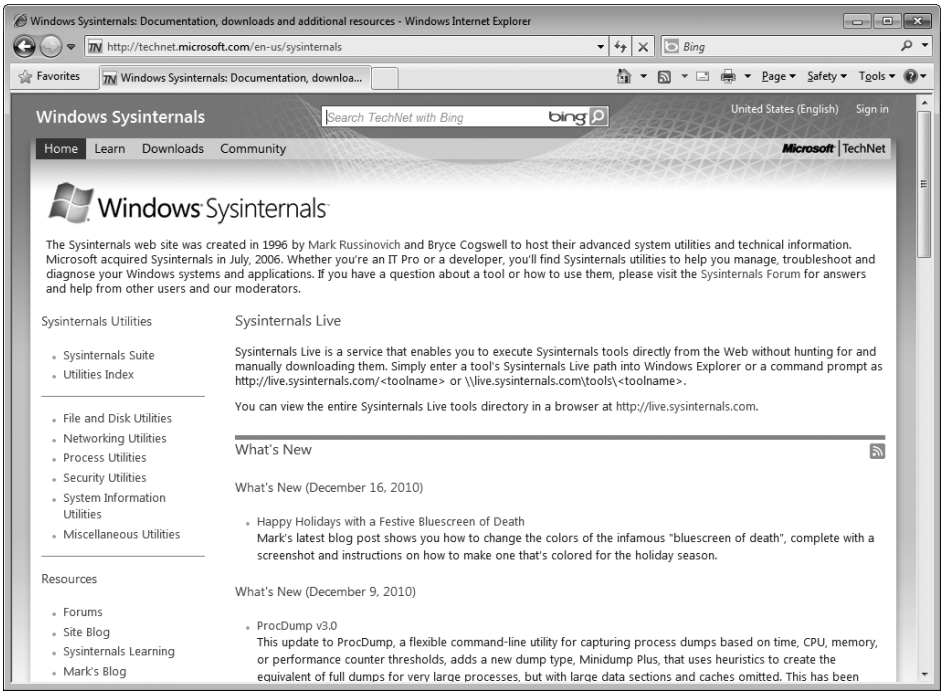


Рис. 1-1. Сайт Windows Sysinternals

Скачивание утилит

Утилиты Sysinternals можно скачивать по отдельности либо полным комплектом в архиве (.zip) под названием *Sysinternals Suite*. На домашней странице Sysinternals есть ссылки на отдельные утилиты. Также имеются ссылки, разбитые по категориям, например, «Файловые и дисковые утилиты» или «Сетевые утилиты», которые ведут на страницы со списками утилит из данной категории.

Каждая загрузка упакована в zip-архив, содержащий исполняемый файл или файлы, текстовый файл лицензионного соглашения с конечным пользователем и, в отдельных случаях, файл электронной справки.

Примечание Утилиты PsTools доступны для скачивания только в виде пакетов — пакета PsTools либо полного пакета Sysinternals Suite.

Мой соавтор Аарон обычно создает папку C:\Program Files\Sysinternals и распаковывает в нее содержимое пакета Sysinternals Suite. Там утилиты невозможно изменить без прав администратора. Затем он добавляет этот каталог в системную переменную окружения Path, что позволяет легко запускать утилиты откуда угодно, в том числе через меню **Start | Run (Пуск | Выполнить)**, как показано на рис. 1-2.

Рис. 1-2. Запуск Procmon с использованием переменной окружения Path и меню Start

«Разблокировка» ZIP-архивов перед извлечением файлов

Перед извлечением содержимого из загруженных ZIP-файлов необходимо сначала удалить метку, который сообщает Windows, что содержимое архива ненадежно, чтобы не выводились предупреждения и сообщения об ошибках, показанные на рис. 1-3 и 1-4. Технология управления вложениями Windows Attachment Execution Service добавляет альтернативный поток данных (ADS) к ZIP-файлу, показывая, что он был скачан из Интернета. При извлечении файлов с помощью Windows Explorer альтернативный поток добавляется ко всем извлекаемым файлам.

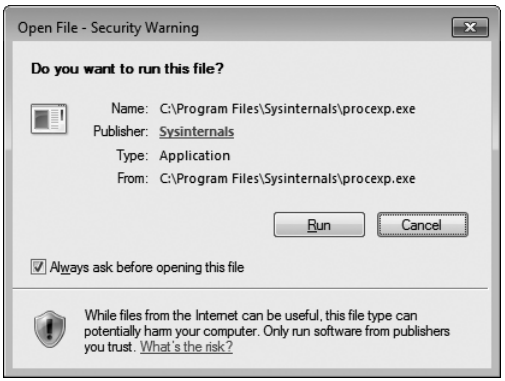


Рис. 1-3. Предупреждение Windows при открытии файлов из Интернета

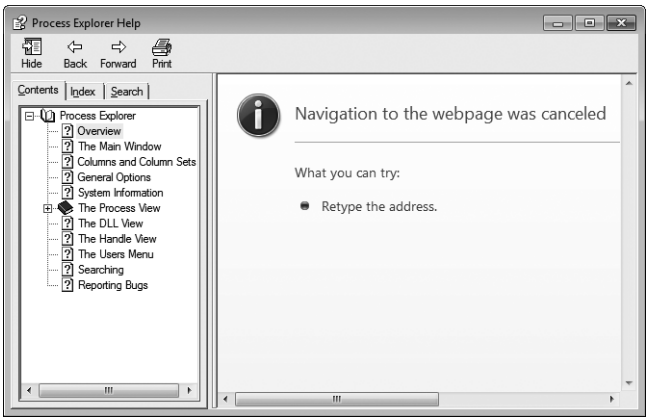


Рис. 1-4. Скомпилированные файлы справки (CHM-файлы) не открываются, так как они отмечены как файлы, полученные из Интернета

Один из способов удаления ADS состоит в том, чтобы открыть окно свойств ZIP-файла в Проводнике Windows и щелкнуть кнопку **Unblock (Разблокировать)** внизу вкладки **General (Общие)**, показанной на рис. 1-5. Второй способ — использование утилиты Sysinternals Streams, описанной в главе 11.

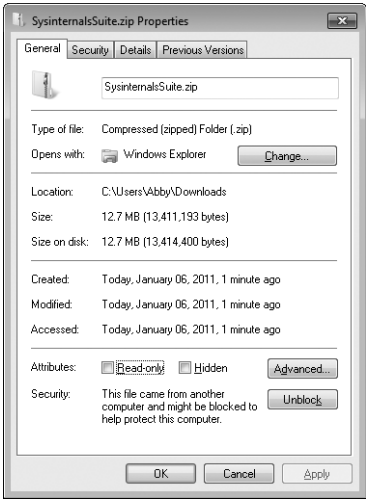


Рис. 1-5. Кнопка Unlock в нижней части окна свойств загруженного файла

Запуск утилит напрямую из Интернета

Sysinternals Live — это служба, позволяющая пользоваться утилитами Sysinternals прямо из Интернета, не загружая и не распаковывая их. Дополнительным преимуществом службы Sysinternals Live является гарантированное получение последних версий программ.

Чтобы запустить утилиту посредством Sysinternals Live в Internet Explorer введите *http://live.sysinternals.com/имя_утилиты.exe* в адресную строку (например, *http://live.sysinternals.com/procmon.exe*). Также можно указать путь для Sysinternals Live в формате UNC: *\\live.sysinternals.com\tools\имя_утилиты.exe* (не забудьте добавить каталог «tools», который не требуется при указании URL). Например, запустить последнюю версию Process Monitor можно так: *\\live.sysinternals.corn\tools\procmon.exe*.

Примечание Синтаксис UNC для запуска утилит через Sysinternals Live требует наличия службы WebClient. В более поздних версиях Windows эта служба может не запускаться автоматически. Ручной запуск службы (например, командой net start webclient) требует прав администратора. Можно запустить службу командой *net use\\live.sysinternals.com* из командной строки или перейдя по адресу *\\live.sysinternals.com* в помощью Windows Explorer.

Можно также сопоставить пути *\\live.sysinternals.com\tools* букву диска или открыть эту папку как удаленный общий ресурс в Проводнике (рис. 1-6). Полный каталог утилит Sysinternals Live доступен в Интернете по адресу *http://live.sysinternals.com*.

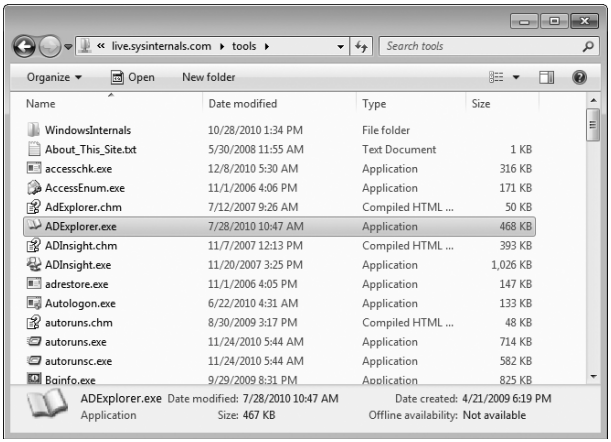


Рис. 1-6. Sysinternals Live в Windows Explorer

Автономные образы

Для облегчения упаковки и распространения все утилиты Sysinternals выпускаются в виде автономных 32-разрядных исполняемых образов, способных запускаться сразу, без установки на ПК. Они содержат все необходимые файлы, извлекая их в папку, в которой находится программа, а если эта папка недоступна для записи (например, если она расположена на носителе, доступном только для чтения), во временную папку текущего пользователя, %TEMP%. Утилиты сами удаляют все извлеченные файлы, когда потребность в них отпадает.

Поддержка и 32- и 64-разрядных систем — это только один из примеров использования технологий автономности утилитами Sysinternals. В 64-

разрядной Windows основная (32-разрядная) программа определяет архитектуру ЦП, извлекает соответствующий двоичный файл для архитектуры x64 или IA64 и запускает его. Например, запустив Process Explorer в системе x64, можно заметить, что процесс Procexp.exe запустил дочерний процесс Procexp64.exe.

Примечание Исполняемые файлы, извлеченные в %TEMP%, не смогут запуститься, если из разрешений %TEMP% удалены разрешения на исполнение.

Большинство утилит Sysinternals, использующих драйвер режима ядра, извлекают файл драйвера в %SystemRoot%\System32\Drivers, загружают драйвер в память и удаляют файл. Образ драйвера остается в памяти до выключения ПК. При запуске новой версии утилиты с обновленным драйвером может потребоваться перезагрузка для загрузки в память нового драйвера.

Форумы Windows Sysinternals

Форумы Windows Sysinternals находятся по адресу *http://forum.sysinternals.com* (рис. 1-7). Это главный и лучший информационный ресурс, где можно получить ответы на вопросы об утилитах Sysinternals и сообщить об ошибках. Возможен поиск сообщений и тем по ключевым словам, так проще узнать, не сталкивался ли кто-нибудь еще с той же проблемой, что и вы. Есть отдельные форумы по каждой из основных утилит Sysinternals, а также форум для предложения идей о новых функциях и утилитах. Кроме того, имеется форум, на котором обсуждается внутреннее устройство Windows, разработка, устранение ошибок и вредоносное ПО.

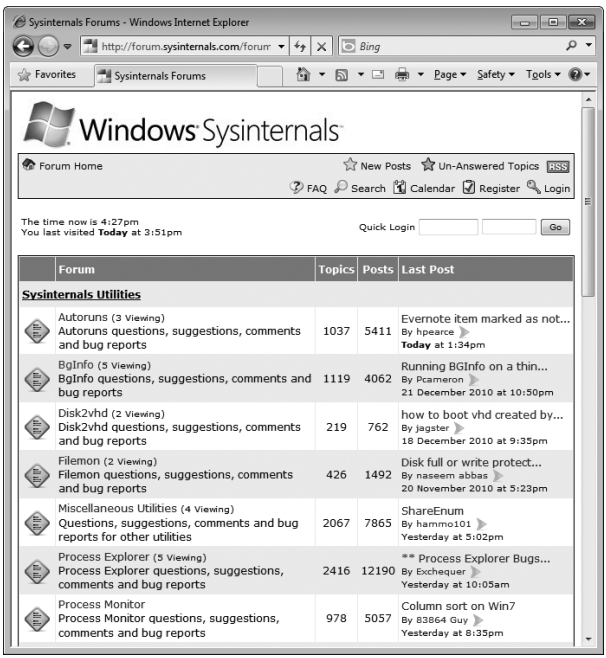


Рис. 1-7. Форумы Windows Sysinternals

Чтобы оставлять сообщения на форумах, необходимо зарегистрироваться и выполнить вход, однако для регистрации требуется лишь необходимый минимум информации. После регистрации можно подписаться на уведомления об ответах в темах и о новых сообщениях на указанных форумах. Можно также отправлять личные сообщения и получать их от других пользователей форума.

Блог сайта Windows Sysinternals

Подписка на блог, посвященный обсуждению сайта Sysinternals — лучший способ получения уведомлений о выходе новых программ, обновлении существующих и о появлении новой информации на сайте. Блог сайта находится по адресу *http://blogs.technet.eom/b/sysinternals*. На главной странице содержится информация об обновлении только основных утилит, в блоге сайта сообщается о всех обновлениях, даже о незначительных.

Блог автора

В блоге М. Руссиновича есть темы о внутреннем устройстве, безопасности и устранению неполадок Windows. Также в блоге есть популярные серии статей о Sysinternals: «The Case of...» — разбор способов решения повседневных проблем с помощью утилит Sysinternals, и «Pushing the Limits» — описание ограничений ресурсов Windows, способов их отслеживания и эффективного использования. Мой блог находится по адресу *http://blogs.technet.eom/b/markrussinovich*. Кроме того, полный список моих сообщений в блоге доступен по ссылке «Mark's Blog» на домашней странице Sysinternals.

Веб-трансляции автора

Полный перечень моих открытых для бесплатного просмотра презентаций на «TechEd» и других конференциях, включая наиболее популярные серии «Case of the Unexplained...» об устранении неполадок средствами утилит Sysinternals, мои интервью телеканалу Channel 9, а также записи программы «Виртуальные круглые столы по Springboard», где я был ведущим, доступны по ссылке «Mark's Webcasts» на домашней странице Sysinternals. Веб-трансляции, записанные к моменту выхода книги, есть на прилагаемом к ней диске.

Лицензирование утилит Sysinternals

Утилиты Sysinternals являются бесплатными. Вы можете устанавливать на свой компьютер и компьютеры вашей компании любое количество копий утилит. Тем не менее, их использование регулируется условиями лицензионного соглашения, которое открывается при запуске программы. Также текст лицензионного соглашения находится на странице Software License, доступной по ссылке с домашней страницы Sysinternals.

Лицензионное соглашение с конечным пользователем и переключатель */accepteula*

Как сказано выше, каждая наша программа требует, чтобы пользователь, запускающий ее на своем компьютере, принял лицензионное соглашение с конечным пользователем (EULA). При первом запуске программы, в том числе консольной, открывается диалоговое окно EULA (рис. 1-8). Пользователь должен щелкнуть кнопку Agree (Принимаю), чтобы запустить программу.

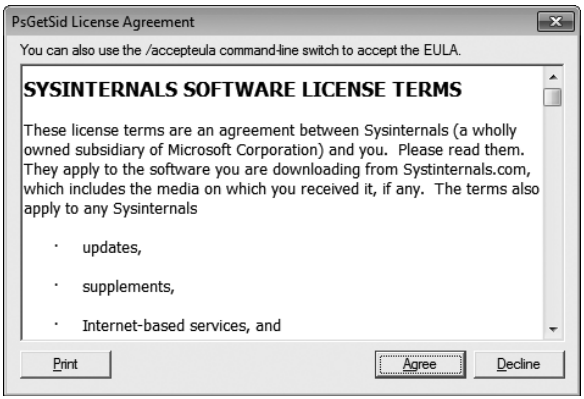


Рис. 1-8. Лицензионное соглашение с конечным пользователем для PsGetSid

Отображение данного окна препятствует использованию программ в сценариях, большинство утилит Sysinternals поддерживает параметр командной строки /accepteula. Использование его также считается согласием с условиями лицензионного соглашения. Например, приведенная ниже команда использует PsExec (см. главу 6) для запуска LogonSessions.exe (см. главу 8) в неинтерактивном режиме на сервере. Параметр /accepteula в командной строке LogonSessions.exe препятствует блокированию исполнения в ожидании щелчка кнопки Agree, которого никогда не произойдет:

```
PsExec \\server1 logonsessions.exe /AcceptEula
```

Имейте в виду, что не все утилиты Sysinternals еще поддерживают параметр /accepteula. Для запуска этих утилит придется вручную подтверждать свое согласие с условиям лицензионного соглашения. Это можно сделать в командной строке, предварительно создав параметр *EulaAccepted* в разделе, который создают утилиты в ветви HKEY CURRENT USER\Software\Sysinternals, например:

```
psexec \\server1 reg add hkcu\software\sysinternals\pendmove /v
eulaaccepted /t reg_dword /d 1 /f
```

Часто задаваемые вопросы по лицензированию Sysinternals

Сколько копий утилит Sysinternals я могу бесплатно скачать и установить на компьютерах моей компании?

Вы можете устанавливать и использовать программное обеспечение на своих компьютерах без ограничений по количеству.

Могу ли я распространять утилиты Sysinternals вместе с моими программами, через мой сайт или журнал?

Нет. Компания Microsoft не предоставляет лицензий на распространение, даже если третья сторона делает это бесплатно. Microsoft призывает скачивать утилиты со своего сайта или запускать их прямо из Интернета, что гарантирует получение самой последней версии утилит.

Могу ли я лицензировать или использовать какой-либо исходный код Sysinternals?

Исходный код Sysinternals больше не доступен для скачивания и лицензирования.

Будут ли инструменты Sysinternals и в дальнейшем распространяться бесплатно?

Да. Microsoft не планирует удалять их или взимать за них плату.

Существует ли служба технической поддержки для пользователей утилит Sysinternals?

Все инструменты Sysinternals предлагаются «как есть», без официальной поддержки Microsoft. Сотрудники компании Microsoft не участвуют в работе форума поддержки сообщества Sysinternals (*http://forum.sysinternals.com*), на котором можно сообщать об ошибках и узнавать о новых возможностях.

Глава 2

Основные понятия Windows

Чем больше вы узнаете о том, как работает Microsoft Windows, тем больше пользы сможете извлечь из утилит Sysinternals. В этой главе дается обзор отдельных понятий Windows, связанных с многочисленными утилитами Sysinternals. На сегодняшний день лучшим и наиболее полным справочником по основным компонентам операционной системы Windows является книга «Windows Internals» (Microsoft Press, 2009)[1]. В этой же книге для таких сложных тем, как управление памятью Windows, вы найдете лишь краткие очерки. Впрочем, эта книга посвящена утилитам Sysinternals, а не Windows вообще, и потому просто не может вместить в себя все подробности, описанные в руководстве «Windows Internals». Нет здесь и полного описания архитектуры Windows, а также разбора некоторых базовых понятий, таких как реестр и отличия TCP от UDP, с которыми читатель, скорее всего, уже знаком.

В этой главе рассматриваются:

- права администратора и запуск программы с правами администратора в различных версиях Windows (относится к большинству утилит);
- процессы, потоки и задания (Process Explorer, Process Monitor, PsTools, VMMap, ProcDump, TCPView, RAMMap);
- пользовательский режим и режим ядра (Process Explorer, Process Monitor, Autoruns, VMMap, ProcDump, DebugView, LiveKd, TCPView, RAMMap, LoadOrder);
- описатели (Process Explorer, Handle);
- стеки вызовов и символы: определения этих понятий и способы настройки символов в утилитах Sysinternals (Process Explorer, Process Monitor, VMMap);
- сеансы, оконные станции, рабочие столы и оконные сообщения (Process Explorer, Process Monitor, PsExec, Adlnsight, Dektops, LogonSessions, WinObj, RegJump).

---

[1] Последним на сегодняшний день является 5-е издание «Windows Internals» (Марк Руссинович, Дэвид Соломон и Алекс Ионеску, Microsoft Press, 2009). 6-е издание тех же авторов на момент написания данной главы готовится к выходу.

Права администратора

В Windows NT исходно реализована полнофункциональная модель управления доступом для защиты уязвимых системных ресурсов от несанкционированного доступа и изменений. В рамках этой модели учетным записям пользователей обычно присваиваются права администратора или права пользователя. Администраторы имеют полный и неограниченный доступ к компьютеру и всем его ресурсам, в то время как пользователям запрещено внесение изменений в конфигурацию операционной системы и доступ к данным других пользователей. Несмотря на это, исторически сложилось так, что до недавнего времени, пока конечные пользователи компьютеров с Windows не стали сплошь и рядом работать с администраторскими правами, многие не знали о существовании этих разграничений. (И даже сейчас локальная учетная запись пользователя, созданная на компьютере с Windows 7, входит в группу **Администраторы (Administrators)**.)

Примечание Пользователи могут управлять компьютером как администраторы, даже не состоя в группе **Администраторы (Administrators)**, если только у них есть возможность настраивать и вызывать программы, запущенные в контексте безопасности с более обширными правами. К таким лазейкам относится получение контроля над системными файлами и реестром, обычно имеющегося у администраторов и служб (как в случае с группой **Опытные пользователи (Power Users)** до Windows Vista); предоставление пользователям привилегий, равных администраторским, таких как привилегии отладки, захвата во владение, восстановления, загрузки драйвера либо включение политики установщика Windows AlwaysInstallElevated («Всегда производить установку с повышенными привилегиями»), при которой любой пользователь вызывает любые MSI-файлы под системной учетной записью.

В последнее время организации, стремящиеся к повышению уровня безопасности и сокращению расходов, стали лишать конечных своих пользователей администраторских прав. После внедрения контроля учетных записей пользователей (UAC) в Windows Vista большинство программ, запускаемых пользователями, в том числе, членами группы **Администраторы**, стало выполняться с правами пользователя, а не администратора. Иногда все же возникает необходимость запустить программу с правами администратора. Раньше (в Windows XP) многие просто не знали, как сделать это, теперь (в Windows Vista) эта процедура претерпела значительные изменения.

Одни утилиты Sysinternals всегда требуют прав администратора, другие полноценно работают без них, третьи же выполняют большинство задач при наличии стандартных прав пользователя, но для выполнения отдельных функций требуют прав администратора. Иными словами, при наличии лишь стандартных прав пользователя они работают в режиме ограниченной функциональности.

Запуск программы с правами администратора в Windows XP и Windows Server 2003

Если вы входите в систему Windows XP или Windows Server 2003 с учетной записью пользователя — члена группы **Администраторы**, то для запуска утилиты Sysinternals с правами администратора отдельных действий не требуется. *Любая* программа, которую вы запустите, будет иметь полные права администратора.

Если же вы входите под учетной записью, не имеющей необходимых привилегий для запуска той или иной утилиты Sysinternals, то вам понадобится получить права администратора от другой учетной записи. Служба «Вторичный вход в систему» (Seclogon) дает возможность программам начать новый процесс под другой учетной записью пользователя на текущем рабочем столе после ввода других учетных данных. Это можно сделать через диалоговое окно **Запуск от имени (RunAs) Проводника (Explorer)** или с помощью программы командной строки Runas.exe.

Для запуска программы с помощью диалогового окна **Запуск от имени (RunAs)** с правами администратора щелкните правой кнопкой мыши любую программу или ярлык в **Проводнике (Explorer)** или в меню **Пуск (Start)** и выберите из контекстного меню **Запуск от имени (RunAs)**. В диалоговом окне **Запуск от имени (RunAs)** выберите вторую кнопку-переключатель **(Следующий пользователь (The Following User))**, как показано на рис. 2-1, введите имя и пароль учетной записи администратора и щелкните **ОК**. Для того чтобы сделать **Запуск от имени (RunAs)** программой по умолчанию для ярлыка, откройте диалоговое окно **Свойства (Properties)**, щелкните кнопку **Расширенные (Advanced)** и установите флажок **Запуск с другими учетными данными (Run With Different Credentials)**.

Рис. 2-1. Диалоговое окно Windows XP Запуск от имени (RunAs) с выбранной второй кнопкой-переключателем

Для запуска программы с правами администратора с помощью программы командной строки Runas.exe, откройте командную строку и запустите Runas.exe следующим образом:

```
runas /u:username program
```

Например, для запуска Process Monitor (Procmon.exe) под локальной учетной записью администратора введите следующую команду:

```
runas /u:administrator procmon.exe
```

После нажатия Enter Runas.exe запросит пароль вашей учетной записи. Вам нужно будет ввести пароль в строку приглашения, так как Runas.exe не принимает пароль, находящийся в командной строке или переданный в нее по стандартному входному потоку. Для того чтобы сохранить пароль учетной записи, который вы вводите впервые, используйте параметр командной строки */savecred*. При дальнейшем использовании */savecred* под той же учетной записью пароль будет извлекаться, и вам не нужно будет вводить его снова. При регулярном использовании данной операции помните, что обычный пользователь, под чьей учетной записью сохранен пароль администратора, сможет теперь использовать Runas.exe для запуска любой программы без ввода пароля.

Для аутентификации по смарт-карте вместо пароля добавьте в командную строку параметр */smartcard*. Теперь вместо пароля вам нужно будет вводить ПИН-код смарт-карты.

Для получения более подробной информации и советов по использованию диалогового окна **Запуск от имени (RunAs)** см. блог Аарона Маргозиса «RunAs basic (and intermediate) topics» по адресу: *http://blogs. msdn. com/b/ aaron_ margosis/archive/2004/06/23/163229.aspx*.

Если вам нужно запустить утилиту Sysinternals с полными правами администратора, но под иной учетной записью (например, для аутентификации в домене), можно воспользоваться сценарием Аарона Маргозиса MakeMeAdmin. Этот сценарий дважды вызывает Runas.exe для запуска командной строки, которая работает под вашей текущей учетной записью пользователя, но с полными правами администратора. (Имейте в виду, что для этого нужно знать учетные данные для записи администратора.) Подробнее об этом см. в блоге Аарона (ищите пост «MakeMeAdmin — temporary admin for your Limited User account») по адресу: *http://blogs.msdn. com/b/aaron_margosis/archive/2004/07/24/193721.aspx*.

Запуск программы с правами администратора в Windows Vista или более поздних версиях

Windows Vista и контроль учетных записей изменили все, что касается запуска программ с правами администратора. Запуск под учетной записью пользователя стал умолчанием даже для членов группы **Администраторы**.

Если вы выполняете вход в систему Windows Vista или более поздней версии под учетной записью пользователя, являющегося членом группы **Администраторы** (первая учетная запись — единственная, которая по умолчанию дает членство в группе **Администраторы** на компьютерах, не

подключенных к домену) или другой группы с широкими возможностями, например, **Операторы архива (Backup Operators)**, либо получившего привилегии, «эквивалентные администраторским», локальные средства защиты (LSA) создают для пользователя два сеанса входа в систему с различными маркерами доступа для каждого (утилита LogonSessions, которая регистрирует эти сеансы, рассматривается в главе 8). Один из этих маркеров обозначает полные права администратора. При этом все группы и привилегии остаются неизменными. Второй — *отфильтрованный* маркер, практически равноценный маркеру пользователя с отключенным членством в «могущественных» группах и снятыми привилегиями. Этот отфильтрованный маркер создает начальные процессы пользователя, например Userinit.exe и Explorer.exe, и наследуются их дочерними процессами. Запуск процесса с полным маркером пользователя требует повышения привилегий UAC при помощи службы Appinfo. Команда Runas.exe по-прежнему присутствует, но не вызывает службу Appinfo, поэтому она действует не совсем так, как в Windows XP. Если вы запустите программу с помощью Runas.exe и укажете учетную запись администратора, программа на выходном языке будет работать под «пользовательской» версией данной учетной записи.

Повысить привилегии UAC для нового процесса можно одним из следующих способов:

- Исполняемый файл содержит манифест, указывающий на то, что он нуждается в повышении привилегий. Такие манифесты имеются в утилитах Sysinternals с графическим пользовательским интерфейсом, например, в Disk2Vhd и RAMMap, которые всегда требуют повышения привилегий (для просмотра манифеста образа служит утилита Sigcheck, см. главу 8).
- Пользователь напрямую посылает запрос на повышение привилегий работающей программы, например, щелкнув ее правой кнопкой мыши и выбрав в контекстном меню опцию **Запуск от имени администратора (Run As Administrator)**.
- Windows эвристически определяет, что приложение является унаследованной программой установки (определение установщика включено по умолчанию, но может быть отключено политикой безопасности).
- Приложение связано с режимом совместимости или т. н. прокладкой (shim), требующей повышения привилегий.

Если родительский процесс уже запущен с маркером администратора, то дочерний процесс просто наследует этот маркер, и повышение привилегий через UAC не требуется. Обычно консольные утилиты, требующие прав администратора, (например, LogonSessions от Sysinternals) не требуют повышения привилегий UAC. Вместо этого следует запускать их из командной строки или консоли Windows PowerShell, запущенных с повышенными привилегиями.

После срабатывания UAC повышение привилегий можно выполнить тремя способами:

- **Пассивно.** Повышение привилегий происходит без вмешательства конечного пользователя. Данная возможность доступна только членам группы **Администраторы**. По умолчанию в Windows 7 она включена для определенных команд Windows. Ее можно включить для всех запросов на повышение привилегий посредством политики безопасности.
- **С запросом.** Система запрашивает у пользователя согласие на повышение привилегий, отображая диалог, показанный на рис. 2-2. Данный вариант доступен только членам группы **Администраторы** и является для них вариантом по умолчанию (кроме пассивного повышения по умолчанию в Windows 7).
- **С запросом учетных данных.** Система запрашивает у пользователя данные учетной записи администратора (рис. 2-3). Это вариант по умолчанию для всех учетных записей, кроме администратора, и единственный способ повышения привилегий для тех, кто не является администратором. Можно дать эту возможность пользователям с правами администратора через политику безопасности.

Имейте в виду, что повышения привилегий через UAC для обычных пользователей могут блокироваться политикой безопасности. В этом случае пользователь при попытке повышения привилегий получает сообщение об ошибке.

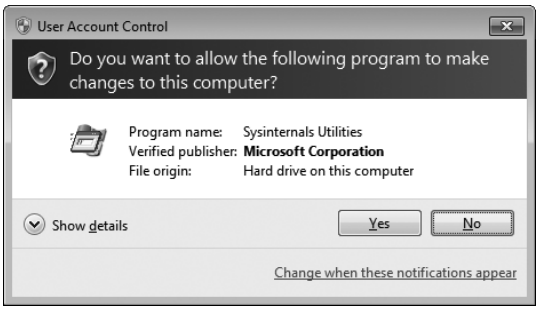


Рис. 2-2. Запрос согласия на повышение привилегий в Windows 7

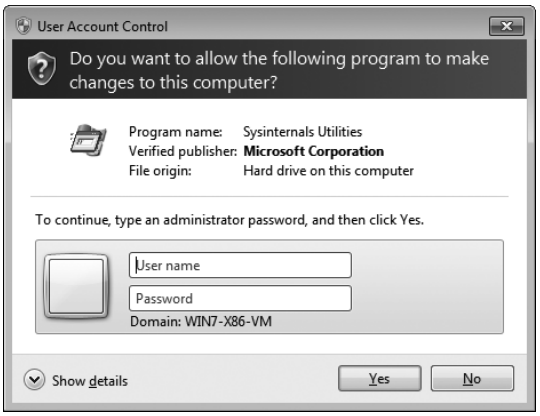


Рис. 2-3. Запрос учетных данных для повышения привилегий в Windows 7

Если включен контроль учетных записей, Windows переходит в режим, аналогичный Windows XP. В этом случае локальные средства защиты не создают отфильтрованные маркеры, и программы, запущенные пользователями — членами группы **Администраторы**, всегда работают с правами администратора. Обратите внимание на то, что отключение UAC приводит также к отключению защищенного режима Internet Explorer, так что Internet Explorer работает с полными правами пользователя, выполнившего вход в систему. Кроме того, отключение UAC приводит к отключению виртуализации файла и реестра этой программы, то есть, функции, позволявшей в Windows XP запускать многие приложения, обычно требующие администраторских прав, под учетной записью пользователя.

Процессы, потоки и задания

На первый взгляд понятия «программа» и «процесс» похожи, но между ними есть принципиальное различие. Программа — это статическая последовательность команд, в то время как процесс представляет собой контейнер для набора ресурсов, используемых для выполнения программы. На самом высоком уровне абстракции процесс Windows включает в себя следующее:

- Уникальный идентификатор, который называется идентификатором процесса (PID).
- Не менее одного работающего потока. Каждый поток в процессе имеет полный доступ ко всем ресурсам, на которые ссылается его процесс-контейнер.
- Закрытое виртуальное адресное пространство — набор адресов виртуальной памяти, используемых процессом для хранения информации и создания ссылок на данные и код.
- Выполняемую программу — исходно запущенный код и данные, отображаемые в виртуальном адресном пространстве процесса.
- Список открытых описателей различных системных ресурсов: семафоров, коммуникационных портов, файлов и пр.
- Контекст безопасности, который называется маркером доступа, идентифицирующий пользователя, группы безопасности, привилегии, состояние виртуализации UAC, идентификатор сеанса входа LSA и идентификатор сеанса служб терминалов.

Каждый процесс содержит также идентификатор своего родительского процесса, однако при завершении родительского процесса эта информация не удаляется. Следовательно, процесс может ссылаться на несуществующий родительский объект-процесс или даже на другой процесс, которому был присвоен исходный идентификатор родительского объекта. Словом, процесс хранит идентификатор своего родительского процесса только в информационных целях.

В Windows модель процессов дополнена концепцией заданий. Основная функция объекта-задания в том, что он позволяет манипулировать группой

процессов как единым целым. Например, задание может использоваться для одновременного завершения группы процессов вместо поочередного завершения, при этом вызывающему процессу не важно, какие процессы находятся в группе. Кроме того, объект-задание позволяет управлять определенными атрибутами и создает ограничения для процесса или процессов, связанных с заданием. В частности, задания могут налагать ограничения по времени выполнения в пользовательском режиме и переданной виртуальной памяти для каждого процесса или задания в целом. *Инструментарий управления Windows* (WMI) загружает свои провайдеры в отдельные хост-процессы, управляемые заданием, которое ограничивает потребление памяти, а также общее количество хост-процессов провайдера WMI, которые могут работать одновременно.

Как сказано выше, процесс — это всего лишь контейнер. С технической точки зрения выполняются не процессы, а содержащиеся в них потоки. Поток — это объект внутри процесса, который Windows может планировать и выполнить. Он включает в себя следующие основные компоненты:

- Содержимое регистров центрального процессора, т.е. состояние процессора. Оно включает указатель, идентифицирующий следующий машинный код, который будет выполнен потоком.
- Два стека — для режима ядра и режима пользователя.
- Закрытая область памяти, которая называется локальной памятью потока (TLS). Она предназначена для использования подсистемами, библиотеками исполняющей системы и динамически подключаемыми библиотеками (DLL).
- Уникальный идентификатор, который называется идентификатором потока (TID). Идентификаторы процессов и потоков генерируются в одном пространстве имен, поэтому они никогда не перекрываются.
- Иногда потоки имеют собственный контекст безопасности, например в случае многопоточных серверных приложений, олицетворяющих контекст безопасности клиентов, которых они обслуживают.

Несмотря на то, что потоки имеют собственный контекст выполнения, все потоки внутри процесса сообща используют его виртуальное адресное пространство (и остальные ресурсы, принадлежащие процессу). Это означает, что все потоки процесса могут читать и записывать память друг друга. Потоки, однако, не могут ссылаться на адресное пространство другого процесса, пока другой процесс не откроет часть своего закрытого адресного пространства в виде общего раздела (в Windows API это называется объектом «проекция файла») или пока один процесс не получит право открыть другой процесс для вызова межпроцессных функций для работы с памятью.

По умолчанию потоки не имеют собственного маркера доступа, однако могут приобрести его. Так отдельным потокам удается олицетворять контекст безопасности, отличный от контекста остальных потоков данного процесса, например, на удаленных Windows-системах, не влияя на другие потоки «своего» процесса.

Пользовательский режим и режим ядра

Для предотвращения доступа к важнейшим данным операционной системы и их модификации пользовательскими приложениями Windows использует два режима доступа к процессору: пользовательский режим и режим ядра. Все процессы, кроме системных, работают в пользовательском режиме («кольцо 3» в терминах архитектуры Intel x86 и x64), в то время как драйверы устройств и компоненты операционной системы, в частности, исполнительная система и ядро, работают только в режиме ядра. Режим ядра соответствует т.н. «кольцо 0» процессоров x86 и x64, поэтому в данном режиме доступна вся память системы и все команды процессора. Разрешая только низкоуровневым компонентам операционной системы с более высоким уровнем привилегий, чем у процессов пользовательского режима, процессор предоставляет разработчикам операционных систем необходимые им возможности, не позволяя приложениям, содержащим ошибки, нарушать стабильность системы в целом.

Примечание Не путайте различие между пользовательским режимом и режимом ядра с различием между правами пользователя и правами администратора. В данном контексте «пользовательский режим» не означает «режим с правами обычного пользователя».

Несмотря на то, что каждый процесс Windows имеет собственное закрытое пространство памяти, операционная система в режиме ядра и код драйверов устройств сообща используют одно виртуальное адресное пространство, которое также включено в адресное пространство каждого процесса. Операционная система помечает каждую страницу виртуальной памяти, указывая, в каком режиме должен работать процессор для чтения или записи этой страницы. Получить доступ к страницам в системном пространстве можно только в режиме ядра, а все страницы пользовательского адресного пространства доступны в пользовательском режиме.

Потоки процессов пользовательского режима переключаются в режим ядра при вызове системных служб. Например, вызов API-функции Windows *ReadFile* в конечном итоге должен вызвать внутреннюю функцию Windows, фактически осуществляющую чтение данных из файла. Так как эта функция имеет доступ к внутренним структурам данных системы, она должна работать в режиме ядра. Переход из пользовательского режима в режим ядра выполняется с помощью специальной команды процессора, которая переводит процессор в режим исполнения системных функций в режиме ядра. Операционная система выполняет соответствующую внутреннюю функцию, которой, в случае с *ReadFile*, является функция ядра *NtReadFile*. Служебные функции ядра проверяют параметры и выполняют соответствующие проверки прав доступа с помощью Монитора защиты (Security Reference Monitor) перед тем как выполнить запрашиваемую операцию. После завершения функции операционная система переключает процессор обратно в пользовательский режим.

Таким образом, для потоков из процессов пользовательского режима характерна попеременная работа в пользовательском режиме и в режиме ядра. Более того, вследствие того, что большинство кода графической и оконной подсистем работает в режиме ядра, процессы приложений, интенсивно использующих графику, могут проводить больше времени в режиме ядра, чем в пользовательском режиме. Посмотреть эти два режима можно на графике использования центрального процессора программы Process Explorer: красная часть графика обозначает время, проведенное в режиме ядра, а зеленая часть графика — время, проведенное в пользовательском режиме.

Описатели

Код ядра Windows, содержащийся в файле в Ntoskrnl.exe и работающий, естественно, в режиме ядра, состоит из ряда подсистем, таких как диспетчер памяти (Memory Manager), диспетчер процессов (Process Manager), диспетчер ввода-вывода (I/O Manager) и диспетчер конфигураций (Configuration Manager) (реестр), являющихся компонентами исполняющей системы. Каждая из этих подсистем определяет с помощью диспетчера объектов один или несколько типов для ресурсов, предоставляемых приложениям. Например, Диспетчер конфигураций (Configuration Manager) определяет объект *раздел* (Key) для представления открытого раздела реестра; диспетчер памяти (Memory Manager) определяет объект-раздел (Section), представляющий общую память; исполняющая подсистема определяет такие объекты, как *семафор*, *мутант* (таково внутрисистемное имя мьютекса) и синхронизирующие объекты-события (объекты-оболочки базовых структур данных, определяемых подсистемой ядра операционной системы). Диспетчер ввода-вывода определяет объект *файл* (File) для представления открытых экземпляров ресурсов драйверов устройств, включающих файлы файловой системы; а диспетчер процессов создает объекты *поток* (Thread) и *процесс* (Process). В каждой версии Windows добавляются новые типы объектов, например, в Windows 7 таковых 42. Типы объектов в той или иной версии Windows, можно посмотреть, запустив утилиту WinObj (см. главу 14) с правами администратора и перейдя в каталог ObjectTypes в пространстве имен диспетчера объектов.

Если приложение захочет использовать один из этих ресурсов, оно должно сначала вызвать соответствующую API-функцию, чтобы создать или открыть ресурс. Например, функция *CreateFile* открывает или создает файл, функция *RegOpenKeyEx* открывает раздел реестра, а *CreateSemaphoreEx* открывает или создает семафор. Если функция выполняется успешно, Windows создает ссылку на объект в таблице описателей, которая ведется исполняющей системой, и возвращает приложению индекс новой записи в таблице описателей.

Описатель нужен приложению для выполнения над ресурсом различных операций. Для обращения к ресурсам и управления ими приложение

передает значение описателя API-функциям, таким как *ReadFile, SetEvent, SetThreadPriority* и *MapViewOfFile*. Чтобы найти объект, на который ссылается описатель, система просматривает с использованием индекса таблицу описателей в поисках записи, содержащей указатель на объект. В этой записи также хранятся сведения о правах доступа процесса, открывшего объект, что позволяет системе пресекать выполнение процессом несанкционированных операций над объектом. Например, если процесс успешно открыл файл для чтения и попытался использовать тот же описатель для записи, соответствующий вызов завершится неудачей.

Когда объект становится ненужным процессу, последний освобождает свой описатель этого объекта. Обычно это делается путем передачи значения описателя API-функции *CloseHandle*. (Обратите внимание на то, что некоторые диспетчеры ресурсов поддерживают иные функции освобождения ресурсов). После завершения процесса все принадлежащие ему описатели закрываются.

Стеки вызовов и символы

Некоторые утилиты Sysinternals, включая Process Explorer, Process Monitor и VMMap, могут отображать сведения об исполняемом в различные моменты коде, т.е. содержимое стеков вызовов. Привязка символов к модулям в адресном пространстве процесса позволяет получить больше понятной человеку контекстной информации об исполняемом коде, особенно системном коде Windows. Если вы разберетесь со стеками вызовов и символами, а также их настройкой в утилитах Sysinternals, вам будет намного проще изучать поведение процессов и отыскивать первопричины проблем.

Что такое стек вызовов?

Исполняемый код в процессе, как правило, представляет собой набор функций. Для выполнения своих задач функция может вызывать другие функции. После завершения функция возвращает управление той функции, которая ее вызвала.

На рис. 2-4 показан вымышленный пример такой ситуации. Программа MyApp.exe поставляется с библиотекой HelperFunctions.dll. Эта DLL содержит функцию *EncryptThisText*, которая шифрует передаваемый ей текст. После выполнения ряда подготовительных операций *EncryptThisText* вызывает API-функцию Windows *CryptEncryptMessage* в Crypt32.dll. В какой-то момент *CryptEncryptMessage* нужно некоторое количество памяти, и она вызывает функцию выделения памяти *malloc* из Msvcrt.dll. После того как *malloc* выполнила свою работу и зарезервировала необходимое количество памяти, управление возвращается функции *CryptEncryptMessage*. Когда завершится и *CryptEncryptMessage*, управление возвращается функции *EncryptThisText*, вернее, ее команде, следующей сразу после вызова *CryptEncryptMessage*.

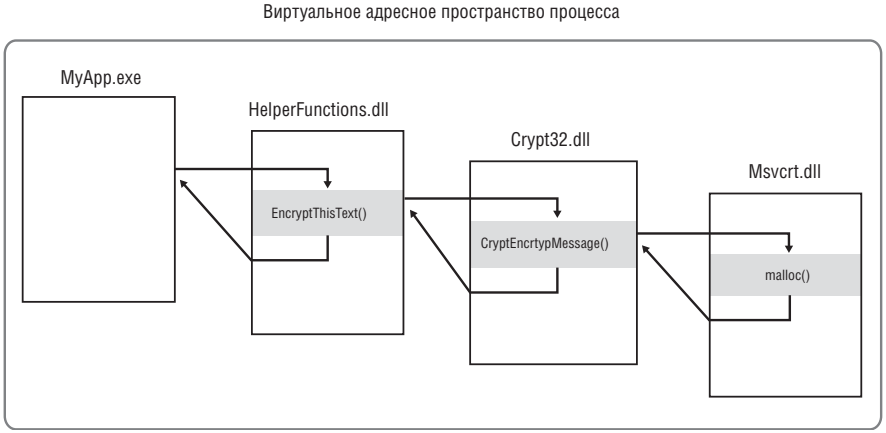


Рис. 2-4. Пример последовательных вызовов

Благодаря стеку вызовов система «знает», какой функции следует возвращать управление после завершения последовательно вызванных функций, как передавать параметры от одной функции к другой и как хранить переменные локальных функций. Стек вызовов работает по принципу «первым вошел, последним вышел», т.е. элементы стека после завершения функций удаляются в порядке, обратном их добавлению при вызовах. Когда функция собирается вызвать другую функцию, она записывает адрес в памяти следующей команды, т.е. команды, которая будет выполняться при завершении вызываемой функции (т.н. «адрес возврата») в верхнюю часть стека. Если вызванная функция, в свою очередь, вызывает третью функцию, та тоже запишет в стек свой адрес возврата. При завершении функции система читает адрес из первого элемента стека и продолжает выполнять код с места, на которое указывает этот адрес.

Элементы стека вызовов формируются по шаблону *имя_модуля!имя_функции+смещение*, где *имя_модуля* — это имя исполняемого файла, содержащего функцию, а *смещение* — это число (в шестнадцатеричной системе счисления) байтов перед началом функции. Если имя функции недоступно, адрес отображается просто как *имя_модуля+смещение*. В показанном выше примере с *malloc* стек вызовов может выглядеть следующим образом:

```
msvc rt!malloc+0x2a
crypt32!CryptEncryptMessage+0x9f
Helper Functions !EncryptThisText+0x43
MyApp.exe+0x25d8
```

В стеке видно, какой код исполняется, и какой путь прошла программа, чтобы запустить его.

Что такое символы?

При проверке начального адреса или адреса возврата потока отладчику легко определить, к какому модулю он принадлежит, просмотрев список за-

груженных модулей и диапазоны их адресов. Но компилятор, конвертируя исходный код разработчика в машинные коды, выбрасывает имена функций. Единственным исключением является ситуация, когда DLL включает таблицу экспорта со списком имен и смещений функций, доступную другим модулям. В таблице экспорта, однако, нет имен внутренних функций библиотеки, равно как и имен точек входа, определяемых во время выполнения.

Примечание Исполняемые файлы, загруженные в процессы пользовательского режима, как правило, являются EXE-файлами, позволяющими запускать новые процессы, либо DLL, загружаемыми в существующий процесс. Бывают исполняемые файлы и с другими расширениями. Так, файлы с расширениями COM или SCR фактически являются EXE-файлами, в то время как файлы с расширениями ACM, AX, CPL, DRV и OCX являются разновидностями DLL. Программы установки обычно распаковывают и запускают EXE-файлы с расширением TMP.

При создании исполняемых файлов компиляторы и компоновщики могут также генерировать соответствующие файлы символов (их расширение по умолчанию — PDB). Файлы символов содержат данные, которые не нужны при запуске исполняемого кода, но могут быть полезны при отладке. Это имена и смещения для точек входа функций, содержащихся в модуле. Имея эту информацию, отладчику легко, взяв адрес в памяти, найти соответствующую функцию по ближайшему предыдущему адресу. В отсутствие символов отладчику приходится пользоваться экспортируемыми функциями (если таковые есть), которые могут и не иметь никакого отношения к коду, который нужно исследовать. В целом, чем больше смещение адреса возврата, тем меньше вероятность того, что вы нашли нужное имя функции.

Примечание Утилиты Sysinternals способны использовать только «родные» (неуправляемые) файлы символов при чтении стеков вызовов. Они не определяют имена функций внутри сборок .NET, созданных с помощью JIT-компилятора.

Символьный файл должен генерироваться одновременно с соответствующим исполняемым файлом, иначе подсистема отладки откажется использовать его. Прежние версии Microsoft Visual C++ создавали файлы символов только для отладочных сборок, пока разработчик не отключал эту функцию вручную. Более новые версии в настоящее время создают символьные файлы и для окончательных сборок, записывая их в одну папку с исполняемыми файлами. Microsoft Visual Basic 6 может создавать символьные файлы, но не делает этого по умолчанию.

Информация в файле символов может быть в разной степени подробной. Полные символьные файлы (иногда называемые закрытыми файлами символов) содержат сведения, отсутствующие в открытых файлах символов, включая путь и номер строки файла исходного кода, в которой определен соответствующий символ, имена и типы параметров функции, а также имена и типы переменных. Компании — производители программного обеспечения, публикующие файлы символов, раскрывают только открытые символьные файлы, оставляя полные (закрытые) символьные файлы для внутреннего использования.

Средства отладки Windows позволяют загружать нужные файлы символов по требованию с веб-сервера. На сервере могут быть символьные файлы для множества версий данного исполняемого файла, и средства отладки загружают именно символы, подходящие к отлаживаемому файлу. (При этом проверяется временная метка и контрольная сумма в заголовке исполняемого файла и уникально его идентифицирующие.)

У Microsoft есть веб-сервер, с которого можно скачать открытые символьные файлы Windows. Установив средства отладки Windows и настроив утилиты Sysinternals для использования сервера символов Microsoft, вы без труда узнаете, какие функции Windows вызывают ваши процессы.

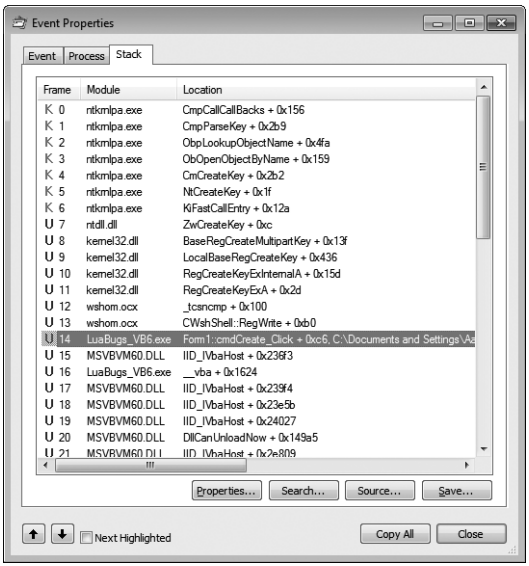


Рис. 2-5. Стек вызовов в Process Monitor с информацией из символьных файлов

На рис. 2-5 показан стек вызовов для события, захваченного утилитой Process Monitor. Присутствие в стеке MSVBVM60.DLL (фреймы 15 и 17-21) указывает на то, что это Visual Basic 6, поскольку MSVBVM60.DLL является библиотекой времени выполнения Visual Basic 6. Большие смещения фреймов MSVBVM60 говорят о том, что для этого модуля символы не доступны, а отображаемые имена не являются реальными именами вызываемых функций. Фрейм 14 отражает вызов функции Forml::cmdCreate. Щелкните главный исполняемый файл (LuaBugs_VB6.exe). Этот фрейм также содержит путь к исходному файлу. Это говорит о том, что перед нами полные данные символов для данного модуля, созданного сторонним разработчиком. Затем эта функция вызывает CWshShelbRegWrite из Wshom.ocx (фрейм 13), значит эта программа, написанная на Visual Basic 6, использует Windows Script Host и ActiveX для записи в реестр. CWshShelkRegWrite вызывает внутреннюю функцию из того же модуля (фрейм 12), а тот — документированную API-функцию Windows *RegCreateKeyExA* из Kernel32.dll (фрейм 11).

Управление передается через внутренние функции Kernel32 (фреймы 8-10) системной API-функции *ZwCreateKey* из Ntdll.dll (фрейм 7). Таким образом, все эти функции были выполнены в пользовательском режиме, о чем свидетельствует буква U в столбце фреймов, однако во фрейме 6 программа переключилась в режим ядра, о чем свидетельствует буква K. Двухбуквенные префиксы функций ядра (фреймы 0-6) обозначают компоненты исполняющей системы, к которым они принадлежат. Например, Cm относится к диспетчеру конфигураций, отвечающему за реестр, а Ob — к диспетчеру объектов. Трассировка стека была записана во время обработки *CmpCallCallBacks* (фрейм 0). Обратите внимание, что символы в фреймах 0-13 были взяты исключительно из открытых символов Windows, загруженных по требованию программой Process Monitor с сервера символов Microsoft.

Настройка символов

Утилиты Sysinternals, использующие символы, требуют два вида информации (рис. 2-6): путь к файлу Dbghelp.dll и путь к символам. Утилиты Sysinternals, которые могут использовать полную символьную информацию для отображения исходных файлов, запрашивают также пути к исходным кодам.

Dbghelp.dll — это одна из библиотек механизма отладки Microsoft. Она обеспечивает поддержку прохода по стеку, загрузки системных файлов и разрешения адресов памяти в имена функций. Только версия Dbghelp.dll, поставляемая со средствами отладки Windows, поддерживает загрузку файлов с серверов символов. Версия Dbghelp.dll, которая поставляется с Windows в каталоге %SystemRoot%\System32, может использовать только локальные символьные файлы. При первом запуске утилиты Sysinternals проверяют пути по умолчанию к средствам отладки и, обнаружив файл Dbghelp.dll, используют его, либо, в противном случае, файл из %SystemRoot%\System32.

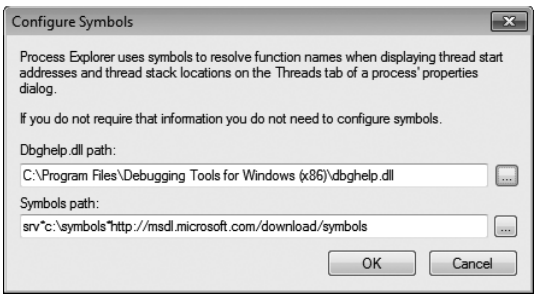


Рис. 2-6. Диалоговое окно настройки символов в Process Explorer

Вот URL для загрузки средств отладки (Debugging Tools) Windows: *http://www.microsoft.com/whdc/devtools/ debugging/defaultmspx*. Раньше установщик средств отладки был доступен для загрузки в виде отдельного файла, но сейчас он входит в пакет Windows SDK. Для использования средств отладки нужно запустить скачанный установщик и выбрать нужные средства отлад-

ки. Пакет содержит свободно распространяемые средства (в виде отдельных файлов для платформ x86, x64 и IA64). Свободно распространяемые пакеты удобны для установки отладчика на другие компьютеры по сети, т.к. они позволяют не запускать установку полного SDK на каждом компьютере.

Путь к символам указывает механизму отладки, где искать символьные файлы, если их нет в каталогах по умолчанию. По умолчанию механизм отладки ищет символьные файлы в каталоге, в котором находится исполняемый файл, а также в каталоге, в котором был исходно сгенерирован символьный файл (если эти каталоги указаны в исполняемом файле).

Путь к символам может включать системные папки и директивы сервера символов. При первом запуске утилиты Sysinternals получают путь к файлам символов из переменной окружения _NT_SYMBOL_PATH. Если эта переменная не определена, используется путь *srv*http://msdl.microsoft.com/download/symbols,* т.е. сервер открытых символов Microsoft, но загруженные символьные файлы при этом не сохраняются в локальном кеше.

Системные папки и директивы сервера символов могут присутствовать в пути одновременно, будучи разделенными точкой с запятой. Очередность просмотра элементов пути определяется их порядком. Так, директивы сервера символов имеют следующий вид: srv*DownstreamStore*SymbolServer, а вот пример пути к файлам символов:

```
C:\MySyms;srv*C:\MSSymbols*http://msdl.microsoft.com/download/symbols
```

Механизм отладки сначала ищет символы в каталогах по умолчанию, затем в C:\MySyms (там удобно хранить закрытые файлы символов созданных вами программ). Если символьный файл не найден, механизм отладки ищет его в C:\MSSymbols, если файла нет и там, он отправляет запрос серверу символов. Если нужный файл есть на сервере символов, механизм отладки загружает его в C:\MSSymbols.

Подробнее о путях, серверах символов, исходных путях и переменных окружения, используемых механизмом отладки, см. в документации по средствам отладки.

Совет Если вам нужны только открытые символы, установите такой путь:

```
srv*c:\symbols*http://msdl.mi crosoft.com/download/symbols
```

В этом случае механизм отладки сначала будет искать символы в C:\Symbols (механизм отладки создает ее, если она не существует). Затем, по мере необходимости, символьные файлы будут загружаться с сервера открытых символов Microsoft. Загруженные файлы записываются в кеш, чтобы не приходилось загружать их снова.

Сеансы, оконные станции, рабочие столы и оконные сообщения

В описании ряда утилит Sysinternals, в том числе Process Explorer, Process Monitor, PsExec, AdInsight, Desktops и LogonSessions, упоминаются сеансы

служб терминалов, идентификаторы сеансов, «консольный сеанс» и «сеанс 0»; интерактивные и неинтерактивные оконные станции, а также прочие программы, запускаемые «с того же рабочего стола». Мало кто знает об этом, но эти понятия могут иметь ключевое значение для решения проблем в Windows.

Для начала рассмотрим схему на рис. 2-7, а затем определимся с терминами. На внешнем уровне находятся сеансы служб терминалов (TS). Каждый сеанс содержит одну или более оконных станций, содержащих рабочие столы. Каждый из этих защищаемых объектов располагает ресурсами, предназначенными для его индивидуального использования. Эти объекты связаны произвольными отношениями с сеансами, которые создаются LSA после локального входа в систему. Хотя в документации Windows не всегда проводится четкое различие между сеансами LSA и сеансами служб терминалов, они являются совершенно разными объектами.

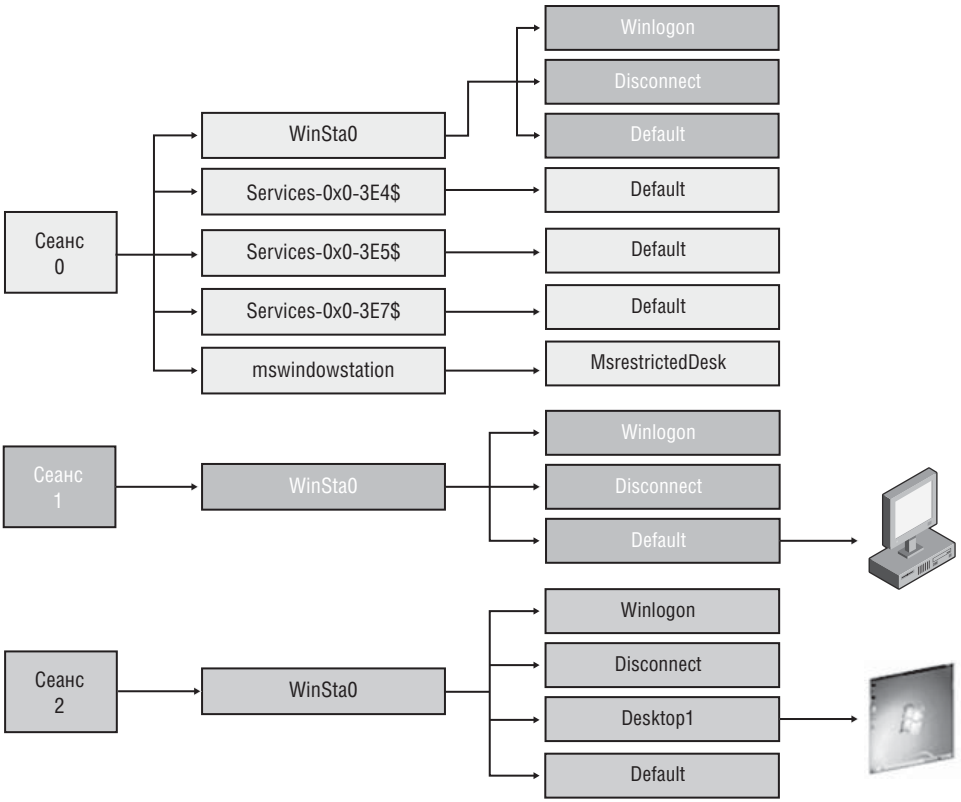


Рис. 2-7. Сеансы, оконные станции и рабочие столы

Сеансы служб терминалов

Службы терминалов поддерживают интерактивные пользовательские сеансы на одном компьютере. Впервые представленные в Windows NT 4.0 Terminal

Server Edition, они не поддерживались клиентскими версиями Windows до Windows XP. Функции, которые они поддерживают, включают быструю смену пользователей (Fast User Switching), удаленный рабочий стол (Remote Desktop), удаленный помощник (Remote Assistance), удаленные локально интегрированные приложения (RAIL или RemoteApps) и функции интеграции виртуальной машины. Важным ограничением клиентских версий Windows (Windows XP, Windows Vista, и Windows 7) является поддержка единственного активного интерактивного сеанса одновременно. Другими словами, только один сеанс может обновлять изображение на дисплее и принимать ввод от клавиатуры и мыши, в то время как разные процессы могут продолжать работу в разных независимых сеансах. Следующее ограничение заключается в том, что подключенный к домену компьютер с Windows XP поддерживает не более одного интерактивного сеанса. Например, если пользователь вошел в систему с консоли, вы можете войти в систему через удаленный рабочий стол, используя ту же учетную запись, и продолжить этот сеанс, однако вы не можете войти под другой учетной записью до тех пор, пока первый пользователь не выйдет из системы.

Сеансы служб терминалов идентифицируются порядковыми числовыми идентификаторами, начиная с сеанса 0. Windows определяет глобальное пространство имен в диспетчере объектов и «локальное» сеансовое пространство имен для каждого сеанса с номером 1 и выше для изоляции сеансов. Глобальное пространство имен служит в качестве локального для процессов сеанса 0 (WinObj дает графическое изображение пространства имен диспетчера объектов, см. главу 14).

Системные процессы и службы Windows всегда работают в сеансе служб терминалов 0. В Windows XP и Windows Server 2003 первый интерактивный пользователь, входящий в систему, также использует сеанс служб терминалов 0 и, следовательно, использует то же локальное пространство имен, что и службы. Windows XP и Windows Server 2003 создают сеансы 1 и далее только по мере необходимости. Если первый пользователь выходит из системы до того, как в нее войдет второй, то второй пользователь также использует сеанс 0. Таким образом, для Windows XP, подсоединенной к домену, сеанс 0 всегда является единственным сеансом.

В Windows Vista и более поздних версиях службы работают в сеансе 0, однако, в целях безопасности, все интерактивные пользовательские сеансы работают в сеансах 1 и выше. Это строгое разделение процессов конечных пользователей и системных процессов называется изоляцией сеанса 0.

Примечание Термин «консольный сеанс» иногда ошибочно считается синонимом сеанса 0. Консольный сеанс — это сеанс служб терминалов, связанный с локально подсоединенными клавиатурой, видео и мышью. Если все активные сеансы системы представляют собой сеансы удаленного рабочего стола, то консольный сеанс остается подключенным и отображает экран входа в систему. Он может являться или не являться сеансом 0 в Windows XP/Windows 2003, однако никогда не является таковым в Windows Vista и более поздних версиях.

Оконные станции

Каждый сеанс служб терминалов содержит одну или несколько именованных оконных станций. Оконная станция — это защищаемый объект, содержащий буфер обмена, атомарную таблицу, а также не менее одного рабочего стола. Каждый процесс связан с одной оконной станцией. В сеансе терминальных служб только оконная станция с именем *WinSta0* может отображать пользовательский интерфейс и принимать ввод от пользователя. В сеансах терминальных служб с номерами 1 и выше Windows создает только оконную станцию WinSta0. (см. рис. 2-8). В сеансе 0 Windows создает, помимо WinSta0, отдельную оконную станцию для каждого сеанса LSA, связанного со службой. Такие сеансы имеют локально уникальный идентификатор (LUID), который добавляется к имени оконной станции. Например, процессы системных служб работают в оконной станции *Service-0x0-3e7$*, в то время как процессы сетевых служб работают в оконной станции *Service-0x0-3e4$*. Эти оконные станции не могут отображать пользовательский интерфейс.

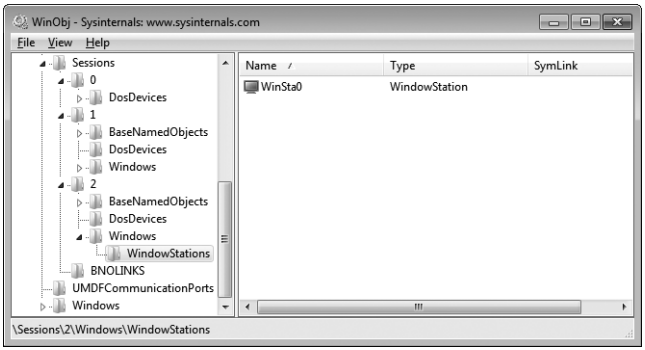


Рис. 2-8. WinObj отображает интерактивную оконную станцию в закрытом пространстве имен сеанса 2

Команда PsExec -s cmd.exe запускает командную строку в оконной станции *Service-0x0-3e7$* и перенаправляет ввод-вывод своей консоли в PsExec (см. главу 6). Параметр -i программы PsExec позволяет задавать сеанс терминальных служб, а также запускает целевой процесс в своей оконной станции WinSta0.

Для службы, настроенной для работы под системной учетной записью, можно также включить параметр **Разрешить взаимодействие с рабочим столом (Allow Service To Interact With Desktop).** При такой настройке служба работает в оконной станции WinSta0 сеанса 0 вместо Service-0x0-3e7$. Когда интерактивный пользователь также находился в сеансе 0, это позволяло службе отображать пользовательский интерфейс конечному пользователю. Оглядываясь назад, следует отметить, что это была не лучшая идея, поэтому компания Microsoft не рекомендует использовать данный метод. Теперь он вовсе не работает из-за изоляции сеанса 0. (Служба обна-

ружения интерактивных служб, UIODetect, частично смягчает негативные последствия этого обстоятельства.)

Рабочие столы

Каждая оконная станция имеет один или более рабочих столов. Рабочий стол — это защищаемый объект с логическим экраном, на котором приложения могут создавать пользовательский интерфейс в виде окон.

Примечание Рабочие столы, описываемые здесь, не имеют отношения к абстракции **Рабочий стол (Desktop)** наверху пространства имен командной оболочки **Проводника (Explorer).**

Пользовательский интерфейс содержат многие рабочие столы, но только один из них может отображаться в данный момент. В интерактивной оконной станции обычно содержатся три рабочих стола: Default, Screen-saver и Winlogon. Default — это рабочий стол, где пользовательские приложения запускаются по умолчанию. Утилита Sysinternals Desktops создает до трех дополнительных рабочих столов для запуска приложений (об этом написано в главе 10). Screen-saver — это рабочий стол, на котором Windows запускает экранную заставку, если включена защита паролем. Winlogon, который также называют защищенным рабочим столом, — это рабочий стол, которому Windows передает управление при нажатии Ctrl+Alt+Del, а также по умолчанию место отображения диалоговых окон повышения привилегий UAC. Разрешения на рабочем столе Winlogon ограничивают доступ только к программам, работающим под системной учетной записью, что обеспечивает безопасность операций, связанных с вводом пароля.

Рис. 2-9. Процесс в сеансе 0 с открытыми описателями объектов рабочего стола и оконной станции

Поскольку процесс связан с оконной станцией, каждый из его потоков связан с рабочим столом внутри оконной станции. Разные потоки процесса могут быть связаны с разными рабочими столами, но, как правило, они связаны с одним рабочим столом.

Некоторые утилиты Sysinternals, в том числе, Process Explorer (описанный в главе 3) и Process Monitor (см. главу 4), опознают идентификатор сеанса службы терминала, к которому принадлежит процесс. Хотя ни одна утилита не может надежно идентифицировать оконную станцию или рабочие столы, с которыми связан процесс, функция Handle View (Просмотр описателей) утилиты Process Explorer может дать информацию об этом в виде открытых описателей оконных станций или объектов рабочих столов. Например, на рис. 2-9 Process Explorer отображает процесс, работающий в качестве системы в сеансе 0 с открытыми описателями рабочего стола \Default и оконной станции \Windows\WindowStations\Service-0x0-3e7$.

Оконные сообщения

В отличие от консольных приложений, GUI-приложения Windows являются событийно-управляемыми. Каждый поток, создающий оконные объекты, имеет очередь, в которую отправляются сообщения. Потоки графического пользовательского интерфейса ожидают, а затем обрабатывают входящие оконные сообщения. Эти сообщения информируют окно о том, что следует сделать или что произошло. Например, сообщения могут передавать окну следующие команды: «обновиться», «переместиться в точку экрана с координатами (x, y)», «закрыться», «была нажата клавиша Enter», «был сделан щелчок правой кнопкой мыши в координатах (x, y)» или «пользователь выходит из системы».

Оконные сообщения отправляются при помощи диспетчера окон. Сообщения могут отправляться любому окну из любого потока, работающего на том же рабочем столе. Диспетчер окон не позволяет программе отправлять оконное сообщение окну с другого рабочего стола. Команды программы Process Monitor /Terminate и /WaitForldle должны отправляться с того же рабочего стола, на котором работает адресат Procmon, поскольку они используют оконные сообщения для передачи существующему адресату команды закрытия, а также определять, готов ли адресат обрабатывать команды в виде оконных сообщений.

Оконные сообщения могут использоваться для имитации действий мыши или клавиатуры. Именно этим занимаются программа RegJump и функция Jump To в Process Monitor и Autoruns, чтобы перейти к разделу в Regedit. Из-за уровней абстракции между физическим нажатием клавиши и появляющимися оконными сообщениями, принимаемыми GUI-программой, конечной программе практически невозможно отличить с абсолютной точностью, нажата ли клавиша на клавиатуре или это другая программа сымитировала нажатие клавиши, отправив оконное сообщение. Это справедливо не только для Windows, но и для других «оконных» систем.

Данная архитектура оконных сообщений (за исключением внедрения многопоточной поддержки 32-разрядных версий Windows), восходит к Windows 1.0, и имеет множество унаследованных свойств. В частности, оконные объекты не имеют дескрипторов защиты и списков управления доступом. Именно поэтому разрешить службам отображение окон на пользовательском рабочем столе было плохой идеей, так как пользовательские программы могли отправлять ошибочные или злонамеренные сообщения окнам системных процессов и, при успешном использовании дыр в защите, управлять этими процессами (это т.н. shatter-атака). Если пользователь не был администратором, повышение привилегий очень простым было делом. Именно в этом заключается главная причина, по которой интерактивные пользователи больше не входят в систему в сеансе 0.

При работе с пользовательскими правами (это режим по умолчанию в Vista и более поздних версиях Windows), а также при повышении привилегий UAC (оно нужно для работы приложений, требующих административских прав, на одном рабочем столе с процессами, обладающими пользовательскими правами), требовалась дополнительная защита от shatter-атак окон процессов с повышенными привилегиями. Результатом стала изоляция привилегий пользовательского интерфейса (UIPI).

До Windows Vista любой процесс, работающий под пользовательской учетной записью, имел полный доступ к управлению любыми другими процессами, работающими под той же записью. При выключенном обязательном контроле целостности (Mandatory Integrity Control, MIC) процессы создаются и работают на определенном уровне целостности (Integrity Level, IL). IL представляет некоторое значение, связанное с относительным уровнем доверия к процессу. Приложения с повышенными привилегиями работают на высоком IL, стандартные пользовательские приложения — на среднем, а процессы с низким уровнем прав (например, Internet Explorer в безопасном режиме) — на низком IL. (Программа Process Explorer, описанная в главе 3, может отображать IL для каждого процесса).

Когда передается оконное сообщение, которое может изменить состояние окна-адресата (например, сообщение о щелчке кнопкой мыши), при активной UIPI диспетчер окон сравнивает IL процесса, отправляющего сообщение, с IL процесса, владеющего окном-адресатом. Если IL отправителя ниже, чем у получателя, сообщение блокируется. По этой причине RegJump и аналогичные функции Jump To должны выполняться с IL не ниже уровня Regedit. Подробнее о MIC и UIPI см. по адресу: *http://msdn.microsoft.com/en-us/library/bb625964.aspx*

Часть II

Работа с утилитами

Глава 3

Process Explorer

Процесс — это базовая сущность для любого компьютера, работающего под управлением Microsoft Windows. Зная процессы, запущенные в данный момент, легко разобраться в утилизации времени центрального процессора (ЦП) и других ресурсов, диагностировать неполадки и выявлять вредоносное ПО. Чаще всего с сайта Sysinternals скачивают Process Explorer и, в чем вы вскоре убедитесь, не без оснований.

Чтобы пользователям было проще разобраться в действиях процессов, работающих на их компьютерах, в Windows с самых первых версий присутствовал компонент Task Manager (Диспетчер задач) — простое приложение для просмотра процессов (приложений и служб), работающих в системе. Чтобы не перегружать пользователей избыточной информацией, Диспетчер задач предоставляет ограниченные сведения. То есть пользователи видят простые списки процессов, служб, пользователей, графики параметров производительности системы и активности сети, а также абстрактный перечень «Приложения» (по сути, список видимых окон в текущем сеансе пользователя). К Диспетчеру задач пользователи обычно прибегают, чтобы узнать, почему компьютер медленно работает и, при необходимости, завершить сбойные процессы. Зачастую его информации недостаточно для выяснения причин сбоя процесса и распознавания вредоносного ПО.

Еще на заре Sysinternals мы с Брюсом Когсвеллом создали множество утилит для заполнения пробелов в данных, выводимых Диспетчером задач. Эти утилиты отслеживали разнообразные аспекты процессов и служб Windows, регистрируя более подробные сведения. Первые три утилиты — PsList, DLLView, и HandleEx (сейчас — просто Handle), положили начало инструментарию Sysinternals для получения подробной информации о процессах. Все они доступны и сейчас, но речь о них пойдет в других главах. Однако вскоре стало ясно, что нужна более удобная программа с единым графическим интерфейсом, чтобы докопаться до сути происходящего в Windows посредством анализа процессов. Так появился Process Explorer (кратко — Procexp).

Обзор Procexp

Procexp несомненно является наиболее функциональной из всех утилит Sysinternals и затрагивает больше аспектов внутреннего устройства Windows, чем остальные (см. главу 2). Вот лишь некоторые основные функции Procexp:

- отображение дерева родства процессов;
- цветовая подсветка различных типов процессов, таких как: службы, процессы .NET, процессы, работающие под той же учетной записью пользователя, что и Procexp, процессы, являющиеся частью задания, и упакованные образы;
- отображение в подсказках информации о командной строке и других процессах;
- выделение новых и недавно завершившихся процессов;
- более точное отображение утилизации ЦП, благодаря чему процессы, потребляющие менее 1% времени ЦП, не отображаются как бездействующие;
- более точные показания загруженности ЦП на основе счетчиков циклов ЦП и переключений контекста;
- дублирование функций Диспетчера задач, который Process Explorer вполне заменяет;
- возможность определить процесс, которому принадлежит то или иное видимое окно на рабочем столе;
- возможность найти окно, принадлежащее указанному процессу, активировать его или закрыть;
- идентификация всех DLL и проецируемых файлов, загруженных процессом, а также всех описателей объектов ядра, открытых процессом;
- поиск процессов с открытыми описателями к объектам ядра, в частности, файлам и папкам;
- поиск процессов, загружавших DLL, идентификация ее пути и других атрибутов;
- графическое представление активности ЦП, выделения памяти и активности ввода-вывода как для системы в целом, так и для отдельных процессов;
- подробные метрики использования памяти и активности ввода-вывода;
- подробная информация о контексте защиты процесса;
- подробная информация о конечных точках TCP/IP для процесса;
- просмотр потоков процесса, их начальных адресов и стеков;
- выгрузка процесса.

Procexp отображает информацию о процессах в нескольких режимах. Окно Procexp по умолчанию состоит из дерева процессов (рис. 3-1; подробнее см. ниже). Procexp может разбивать главное окно на верхнюю и нижнюю

панели. На верхней панели находится список процессов, а на нижней — режим просмотра DLL или режим просмотра описателя. С помощью режима просмотра DLL можно более подробно изучать DLL и проецируемые файлы, загруженные процессом, выбранным на верхней панели. Режим просмотра описателя позволяет исследовать все объекты ядра, открытые в данный момент выбранным процессом, включая файлы, папки, разделы реестра, оконные станции, рабочие столы, конечные точки сети и объекты синхронизации, но не ограничиваясь ими. Режимы просмотра DLL и описателей в подробностях рассматриваются ниже в этой главе. Диалоговое окно **Properties (Свойства)** для процесса предоставляет огромное количество информации о конкретном процессе (см. ниже).

Procexp работает в Windows XP и более поздних версиях с архитектурой x86, x64, и IA64, а также в Windows Server 2003 и выше.

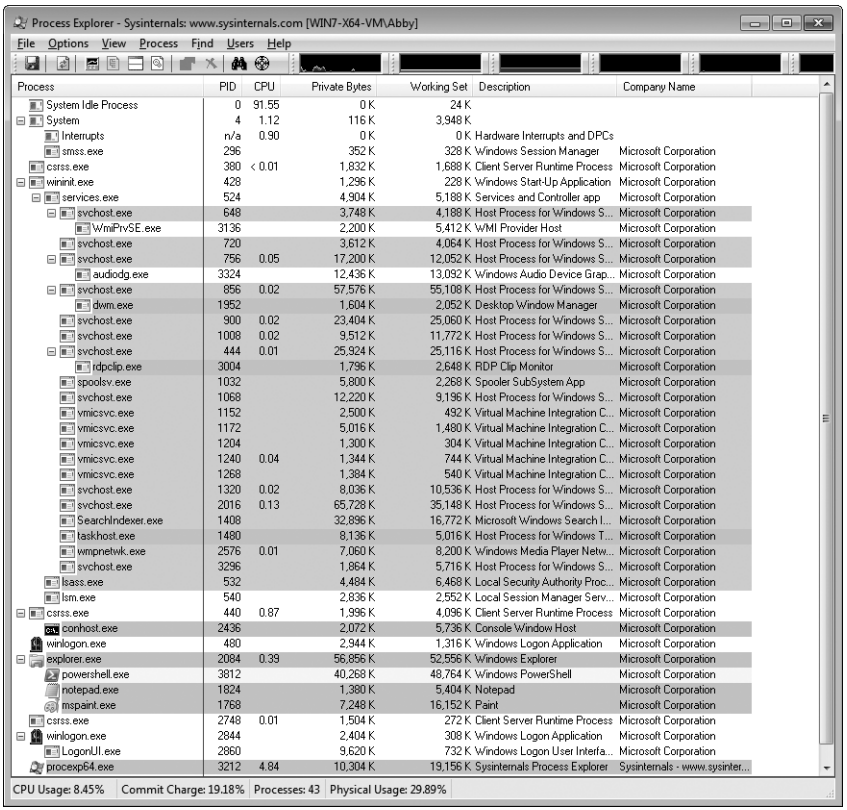


Рис. 3-1. Дерево процессов в Procexp

Мониторинг загруженности ЦП

Ранние версии Windows могли лишь приблизительно измерять реальную загруженность центрального процессора. В момент прерывания, генерируемого часами (в большинстве систем интервал между ними составляет 15,6 мс),

Windows идентифицирует поток, исполняемый в данный момент на каждом ЦП. Если поток исполняется в режиме ядра, его время ядра увеличивается на 15,6 мс, в противном случае увеличивается его время работы в пользовательском режиме (на ту же величину). Возможно, поток успел проработать в течение лишь нескольких тактов ЦП к моменту прерывания, но ему все равно «начисляется» весь интервал длиной 15,6 мс. В течение этого времени могли исполняться сотни разных потоков, но время добавляется только потоку, «застигнутому» прерыванием системных часов в момент исполнения. Диспетчер задач Windows использует эти приблизительные данные в отчетах о загруженности ЦП даже в более поздних версиях Windows, имеющих более точные показатели. Точность данных Диспетчера задач еще больше снижается при округлении до ближайших целых при расчете относительных величин, так что процессы, потоки которых занимают менее 1% времени ЦП, не отличаются от бездействующих процессов. Наконец, Диспетчер задач не учитывает время, затраченное ЦП на обслуживание прерываний и отложенные вызовы процедур (DPC), ошибочно приписывая его процессу System Idle (Бездействие системы).

Procexp отображает загруженность ЦП точнее Диспетчера задач. Во-первых, Procexp по умолчанию показывает загруженность ЦП каждым процессом в процентах с точностью до сотой, а не до целого числа. Во-вторых, Procexp отслеживает время, затраченное на обслуживание прерываний и DPC, не приписывая его процессу System Idle. Наконец, Procexp использует дополнительные системные показатели, что позволяет идентифицировать процессы, слабо загружающие ЦП, и точнее учитывать фактическую загруженность ЦП. В Windows XP, Vista и Windows 7, а также в соответствующих серверных версиях для этого предусмотрены разные счетчики. Procexp использует все доступные счетчики для достижения максимальной точности измерений.

Во всех поддерживаемых версиях Windows все потоки регистрируют свои переключения контекста, т. е. число переключений контекста ЦП, выполненных для начала исполнения потока. Procexp отслеживает изменения соответствующих значений в поле **Context Switch Delta** (**Увеличение числа переключений контекста**). Это позволяет видеть даже процессы, слабо загружающие ЦП. К ним обычно относятся процессы, которые периодически «просыпаются» для поиска изменений состояния и не используют системные механизмы синхронизации (иначе такие процессы могли бы «спать» до фактического изменения состояния). Такие процессы потребляют больше ресурсов, чем нужно, а их код и данные приходится подкачивать в рабочий набор, на что расходуется оперативная память. Если в Windows XP или Windows Vista в столбце **CPU (ЦП)** Procexp показывает значение «< 0.01», процесс вызывает переключение контекста в интервале системного таймера, но загружает ЦП меньше, чем на 0,01% (с учетом округления).

Переключение контекста свидетельствует о том, что поток исполняется, но не несет информации о времени его исполнения. Помимо переключений кон-

текста Windows Vista и более поздние версии измеряют фактическое время ЦП в режиме ядра и пользователя, использованное каждым потоком. Если включить отображение поля **CPU Cycles Delta (Разница числа циклов ЦП)**, Procexp будет отслеживать и выводить эти данные. Если в Windows Vista это поле содержит «< 0.01», соответствующий процесс очень слабо загружает ЦП. Как и в случае с **Context Switch Delta**, столбец **CPU Cycles Delta** помогает выявлять процессы, потребляющих избыточное количество ресурсов.

В Windows Vista утилита Procexp может подсчитывать переключения контекста, но не время ЦП при обслуживании прерываний и DPC. В Windows 7 возможен точный учет времени ЦП, в том затраченного на обработку прерываний и DPC. Таким образом, в Windows 7 Procexp отображает относительную загруженность ЦП в процентах, рассчитанную по фактически затраченному времени ЦП, а не по неточным данным таймера Windows. Для процессов, слабо загружающих ЦП, отображается значение «<0.01», выводить дополнительные данные для них не требуется. Procexp вычисляет загруженность ЦП гораздо точнее Диспетчера задач. В результате оказывается (порой неожиданно), что ЦП загружен куда сильнее, чем по данным Диспетчера задач.

Права администратора

Procexp не всегда требует наличия прав администратора, однако значительная часть сведений о системе доступна только при работе с повышенным уровнем разрешений. Особенно это касается процессов, работающих вне текущего сеанса пользователя. Здесь возможности Procexp определяются наличием привилегии Debug Programs (Отладка программ), предоставляемой по умолчанию группе Администраторы. Там, где политика безопасности не дает администраторам привилегию Debug Programs, использовать все возможности Procexp не удастся. Procexp отображает максимум доступной информации, если же это невозможно, программа оставляет поля пустыми или выводит в них «access denied» (в доступе отказано).

Примечание В Windows Vista и выше даже полных прав администратора недостаточно для чтения подробной информации о защищенных процессах. Audiodg.exe и System являются защищенными процессами, о чем Procexp сообщает на странице **Security (Безопасность)** диалогового окна процесса **Properties**.

Если вы не вошли в систему под учетной записью администратора, то для запуска Procexp с правами администратора в Windows XP и Server 2003 вам нужно запустить программу командой **Run As (Запуск от имени)** или же из другой программы (например, cmd.exe), которая уже работает под учетной записью администратора. В Windows Vista и более поздних версиях для этой цели служит параметр **Run As Administrator (Запуск от имени администратора)**.

В Windows Vista и выше Procexp предлагает два дополнительных варианта. Если Procexp работает без повышенных привилегий, то, выбрав **Show**

Details For All Processes (Показать подробности для всех процессов) в меню **File (Файл)**, можно перезагрузить Procexp с повышением привилегий через UAC. Второй вариант — запуск Procexp с параметром */e*, для чего также требуется повышение привилегий через UAC. (Разумеется, это возможно, если вы работаете в контексте, допускающем повышение привилегий; подробнее см. в главе 2).

Главное окно

Список процессов представляет собой таблицу, в которой каждая строка представляет процесс в системе, а столбцы (поля) — постоянно обновляющиеся атрибуты этих процессов. Можно выбирать атрибуты для отображения, изменять размер и порядок сортировки столбцов, а также сохранять настройки столбцов для дальнейшего использования. На панели инструментов Procexp имеются средства для выполнения общих операций и построения графиков, отражающих значения заданных параметров для всей системы. В строке состояния отображается значение параметров, выбранных пользователем. Ниже мы остановимся на этих функциях подробнее.

Список процессов

Каждая строка в списке процессов представляет процесс, исполняемый на локальном компьютере, хотя это не совсем корректная фраза. Как любит подчеркивать мой друг и соавтор Дэвид Соломон, процессы не исполняются, исполняться могут только потоки. Именно потоки, а не процессы являются теми объектами, которые Windows планирует к исполнению, и которые потребляют время ЦП. Процесс — это всего лишь контейнер для набора ресурсов, в том числе для потоков. Также не совсем корректно говорить «активные процессы» или «процессы с работающими потоками», поскольку большая часть жизненного цикла многих процессов проходит при отсутствии работающих или запланированных для исполнения потоков. Следовательно, каждая строка списка процессов на самом деле отображает объект процесса в системе, который имеет собственное виртуальное адресное пространство, а также один или несколько потоков, которые, возможно, исполняют код. Кроме того, как будет сказано ниже, первые три строки в режиме просмотра по умолчанию (режим дерева) являются исключениями. Забегая вперед, скажу, что буду называть их именно «работающими процессами».

Подсветка процессов

Одна из важных функций списка процессов — разноцветная подсветка, выделяющая различные типы процессов. Можно самостоятельно задавать цвет подсветки для каждого типа процессов, по умолчанию установлены следующие цвета:

- **Голубой** Процессы, работающие под той же учетной записью пользователя, что и Procexp (т. н. «свои процессы»). Учтите, что, несмотря на это, процессы могут находиться в разных сеансах LSA, на разных уровнях

целостности (IL) и сеансах терминалов. Поэтому процессы, запущенные из-под одной учетной записи могут функционировать в разных контекстах защит. Обратите внимание и на то, что при запуске Procmon из-под другой учетной записи остальные приложения, запущенные в этом сеансе, не будут подсвечиваться как «свои».

- **Розовый** Службы (процессы-контейнеры служб Windows).
- **Фиолетовый** Упакованные образы. Procexp с помощью простых эвристических процедур определяет исполняемые файлы, содержащие код в сжатом и (или) зашифрованном виде. Зачастую вредоносное ПО использует данную технологию, чтобы спрятаться от антивирусных программ, а затем распаковать свой код в памяти и выполнить его. Обратите внимание на то, что иногда эвристические процедуры вызывают ложные срабатывания. Чаще всего это происходит с отладочными сборками Microsoft Visual C++.
- **Коричневый** Задания (процессы, связанные с заданиями). Задание — это объект Windows, позволяющий управлять процессами как единым целым. На задания могут налагаться ограничения, например по выделенной памяти или времени исполнения. Процесс может входить не более чем в одно задание. По умолчанию задания не подсвечиваются.
- **Желтый** Процессы .NET (процессы, использующие Microsoft .NET Framework).
- **Темно-серый** Приостановленные процессы. В этих процессах все потоки приостановлены и не могут быть запланированы для исполнения. Процессы, завершившиеся аварийно, могут отображаться как приостановленные, в отчетах об ошибках Windows при этом указывается аварийное завершение (не путайте темно-серый цвет со светло-серым, которым в цветовых схемах Windows по умолчанию подсвечивается выбранная строка, когда окно Procexp не активно).

Если процесс относится к нескольким категориям, цвет подсветки определяется с учетом принятых приоритетов: приостановленные, упакованные, .NET, задания, службы, «свои» процессы. Например, если процесс содержит службу и использует .NET Framework, Procexp подсвечивает его как .NET-процесс, так приоритет этой категории выше, чем служб. Procexp требует прав администратора для распознавания упакованных образов, .NET-процессов и принадлежности к заданию, а также если процесс работает на более высоком IL, чем Procexp, или под другой учетной записью.

Помимо типов процессов в Procexp подсветкой выделяются новые и недавно завершенные процессы. По умолчанию новые процессы подсвечиваются зеленым в течение секунды. Когда процесс завершается, Procexp сохраняет его в списке, на секунду подсвечивая красным. Имейте в виду: хотя подсвеченный красным процесс еще отображается в списке, он уже завершился и больше не существует. Можно настроить длительность подсветки, выбрав пункт **Difference Highlight Duration (Длительность подсветки)** в

меню **Options (Параметры)** и введя число от 0 до 9 (рис. 3-2). Учтите также, что фактическая длительность подсветки зависит от интервала обновления окна Procexp: новые и завершившиеся процессы подсвечиваются только по истечении интервала обновления.

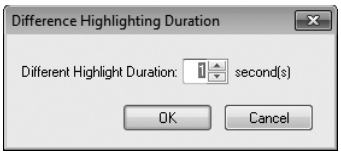


Рис. 3-2. Диалоговое окно Difference Highlight Duration

Для изменения цвета подсветки выберите команду **Configure Highlighting (Настройка подсветки)** в меню **Options**. Как показано на рис. 3-3, можно включать и отключать подсветку, помечая соответствующие флажки. Флажки **New Objects (Новые объекты)** и **Deleted Objects (Удаленные объекты)** также связаны с элементами, отображаемыми в режиме просмотра DLL и режиме просмотра описателей. Флажок **Relocated DLLs (Перемещенные DLL)**, не помеченный по умолчанию, действует только в режиме просмотра DLL. Щелкните кнопку **Change (Изменить)** — откроется окно **Color-picker (Палитра)**, и выберите цвета подсветки соответствующего типа процессов (цвет подсветки приостановленных процессов изменить невозможно).

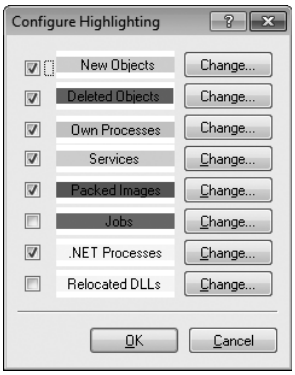


Рис. 3-3. Диалоговое окно Configure Highlighting

Обновление окна

По умолчанию Procexp обновляет отображаемые динамические атрибуты ежесекундно. Динамические атрибуты — это атрибуты, которые периодически изменяются, например, затраченное время ЦП. Чтоб приостановить обновление, нажмите пробел. После повторного нажатия пробела автоматическое обновление возобновляется. Чтобы однократно обновить все отображаемые данные (динамические и статические), нажмите F5 или щелкните значок **Refresh (Обновить)** на панели инструментов. Изменять интервал автоматического обновления можно с помощью пункта **Update Speed**

(Скорость обновления) в меню **View (Вид)**. Длительность интервала обновления — от 0,5 до 10 с.

Совет Изучать объекты, созданные и удаленные за произвольный промежуток времени, удобно при ручном обновлении и подсветке изменений. Приостановите обновление, выполните нужные действия в системе, а затем вернитесь в Procexp и нажмите F5.

Столбцы по умолчанию

Каждый столбец в списке процессов представляет тот или иной статический либо динамический атрибут процесса. Динамические атрибуты обновляются при каждом автоматическом обновлении. Procexp по умолчанию отображает следующие столбцы:

- **Process (Процесс)** Имя и значок исполняемого файла, если Procexp может идентифицировать полный путь к исполняемому файлу. Первые три строки списка представляют «псевдопроцессы», о которых будет сказано ниже.
- **PID (ID процесса)** Идентификатор процесса.
- **CPU (ЦП)** Время ЦП, затраченное в течение последнего интервала обновления (в процентах, округленное до двух десятичных знаков; подробнее см. ниже в описании вкладки Process Performance; о загруженности ЦП см. выше в этой главе).
- **Private bytes (Закрытая память)** Число байтов памяти, выделенной и переданной процессу для монопольного (а не для совместного с другими процессами) использования. Закрытая память процесса включает его кучу и стек. Постоянный рост данной величины зачастую свидетельствует об утечках памяти.
- **Working Set (Рабочий набор)** Объем физической памяти, выделенный процессу Диспетчером памяти.
- **Description and Company Name (Описание и название компании)** Информация из образа исполняемого файла. Отображается, только если Procexp идентифицирует путь к файлу и сможет прочитать его. Если утилита Procexp работает без прав администратора, она не сможет получить информацию о процессах, не являющихся службами и работающих в другом контексте защиты.

Возможно отображение и множества других атрибутов, о которых рассказывается ниже.

Допускается изменять ширину столбцов, перетаскивая их границы в «шапке» списка. Чтобы включить автоподбор ширины столбца по его содержимому, нужно дважды щелкнуть его границу справа от заголовка столбца. Чтобы изменить порядок столбцов, перетащите их заголовки как требуется (но столбец **Process** всегда будет первым слева, он также всегда будет виден). Столбцы, содержимое которых не умещается в отведенном для него месте, можно прокручивать по горизонтали.

Щелчок заголовка столбца сортирует таблицу по содержимому этого столбца в порядке убывания, повторный щелчок заголовка того же столбца сортирует таблицу по убыванию и т.д. Например, если отсортировать список по убыванию значений в столбце **CPU**, в начале списка будут отображаться процессы, сильнее всего загружающие ЦП. Список автоматически упорядочивается заново при каждой регенерации, так как загрузка ЦП различными процессами периодически увеличивается или уменьшается. Все это, опять же, не относится к столбцу **Process.**

Вот одна из хитростей Procexp: если нажать Ctrl+C в главном окне или на нижней панели, содержимое выбранной строки будет скопировано в буфер обмена как текст.

Дерево процессов

Как сказано выше, столбец **Process** всегда отображается первым. Доступны три режима сортировки процессов: по возрастанию и по убыванию некоторого параметра, а также представление в виде дерева.

По умолчанию Procexp отображает процессы в виде дерева, отражающего «родство» процессов. Как только один процесс порождает другой, Windows вносит ID (PID) процесса-создателя (это т.н. *родительский* процесс) во внутренние структуры данных нового (*дочернего*) процесса. Procexp использует эти данные для построения дерева процессов. В отличие от UNIX, родство процессов не задействовано в системе Windows, поэтому при завершении процесса в его дочерние процессы не записываются данные о процессе-«дедушке». Процессы, у которых нет существующего предка, выравниваются по левому краю столбца.

Можно сворачивать и разворачивать элементы-ветви дерева, щелкая значки «плюс» (+) и «минус» (–) слева от родительского процесса дереве либо выделяя нужные узлы и нажимая клавиши со стрелками влево и вправо. Свернутые узлы остаются свернутыми при смене порядка сортировки в **Process** или любом другом столбце.

Щелкая заголовок столбца **Process**, можно последовательно изменять порядок сортировки имен процессов: по возрастанию, по убыванию, дерево и т.д. Кроме того, можно в любое время включить просмотр дерева, нажав Ctrl+T или щелкнув значок **Show Process Tree (Отобразить дерево процесса)** на панели инструментов.

Всплывающие подсказки

Если навести указатель мыши на поле, содержимое которого не умещается в нем, отображается подсказка с полным текстом поля (столбец **Process** и здесь является исключением).

По умолчанию при наведении указателя мыши на любое имя процесса отображается его командная строка и полный путь к образу исполняемого файла (если Procexp может получить данную информацию: в некоторых случаях для этого требуются права администратора). Командная строка и путь к образу не отображаются в подсказке, если соответствующие столбцы

выводятся на экране. Аналогично, если столбцы **Description** или **Company Name** скрыты, данная информация появится в подсказке.

Если навести указатель на процесс-контейнер служб, в подсказке будет перечень отображаемых и внутренних имен всех служб из этого процесса. Для taskeng.exe в Windows Vista или taskhost.exe в Windows 7 отображается подсказка с перечнем заданий в этом процессе. Если же навести указатель на процесс iexplore.exe, содержащий браузер Internet Explorer 8 и выше с открытыми вкладками, в подсказку будет добавлен текст вкладки: это позволяет определить процесс, в котором находится та или иная веб-страница.

Если у процесса есть пользовательский комментарий, а столбец **Comment** скрыт, комментарий также выводится в подсказке (комментарий задают на вкладке **Image** в окне свойств процесса, подробнее об этом — ниже).

Отображаемые сведения

Некоторые стандартные сведения Procexp показывает на любом стандартном компьютере с Windows. К ним относятся некоторые процессы и их дочерние процессы, а также псевдопроцессы, которые Procexp использует, чтобы различать категории активности в режиме ядра.

Системные процессы Первые три строки в режиме дерева — это **System Idle (Бездействие системы)**, **System (Система)** и **Interrupts (Прерывания)**. **System Idle** и **Interrupts** не являются настоящими процессами операционной системы, а в процессе **System** не исполняется код пользовательского режима.

Процесс **System Idle** (или просто **Idle**) включает по одному «потоку» на каждый ЦП, он служит для учета времени бездействия ЦП, т. е. времени, в течение которого Windows не исполняет код. Поскольку это не настоящий процесс, у него нет PID, так как в Windows на самом деле нет PID 0. И все же Диспетчер задач показывает «искусственный» процесс **System Idle** с идентификатором 0, поэтому Procexp по его примеру присваивает данному процессу PID 0.

Процесс **System** содержит только системные потоки в режиме ядра, работающие, разумеется, в режиме ядра. Обычно эти потоки исполняют код операционной системы из Ntoskrnl.exe и код драйверов устройств.

Псевдопроцесс **Interrupts** — представление времени в режиме ядра, затраченное на обслуживание прерываний и вызовов отложенных процедур (DPC). Procexp отображает **Interrupts** как дочерний процесс **System**, поскольку он использует только время в режиме ядра. Windows не приписывает время, «затраченное» данным псевдопроцессом, процессу **System** и другим процессам. Диспетчер задач ошибочно приписывает время обработки прерываний и DPC процессу **System Idle**. В результате Диспетчер задач отображает системы, обрабатывающие много прерываний, как бездействующие. Если ваша система сильно загружена обработкой прерываний и DPC, желательно выяснить причину этого с помощью утилиты Xperf, отслеживающей прерывания и DPC, либо Kernrate, ведущей

мониторинг загруженности ЦП в режиме ядра. Подробнее о прерываниях и DPC см. в нашей книге Windows Internals («Внутреннее устройство Microsoft Windows», Русская редакция, 2008).

Процессы загрузки и входа в систему С момента загрузки Windows до входа в систему первого пользователя запускается целый ряд процессов. К тому моменту, как вы войдете в систему и увидите дерево процессов в Procexp, некоторые из них уже завершатся. Именно поэтому оболочка (обычно это Проводник — Explorer.exe) появляется в левой части окна как процесс без родителя. Подробнее о событиях во время загрузки и входа в систему также см. в книге «Внутреннее устройство Microsoft Windows».

Процедура загрузки Windows XP и Windows Vista различается. В Windows XP, как показано на рис. 3-4, процесс System запускает Smss.exe (Session Manager, Диспетчер сеансов), а тот запускает Csrss.exe и Winlogon.exe. Далее Winlogon запускает Services.exe (Service Control Manager, Диспетчер управления сервисами), Lsass.exe (подсистему Локальных средств защиты), а также два процесса, которых нет на рисунке. Первый — LogonUI.exe, демонстрирующий экран входа в системах, не присоединенных к домену, второй — Userinit.exe, запускаемый Windows после входа пользователя в систему. Userinit.exe запускает Explorer.exe (оболочку пользователя) и завершается. Большинство пользовательских приложений являются прямыми либо непрямыми потомками Explorer.exe. Процессы служб — почти всегда потомки Services.exe. Заметьте, что в Windows XP процесс Services.exe содержит некоторые службы, и поэтому подсвечивается розовым.

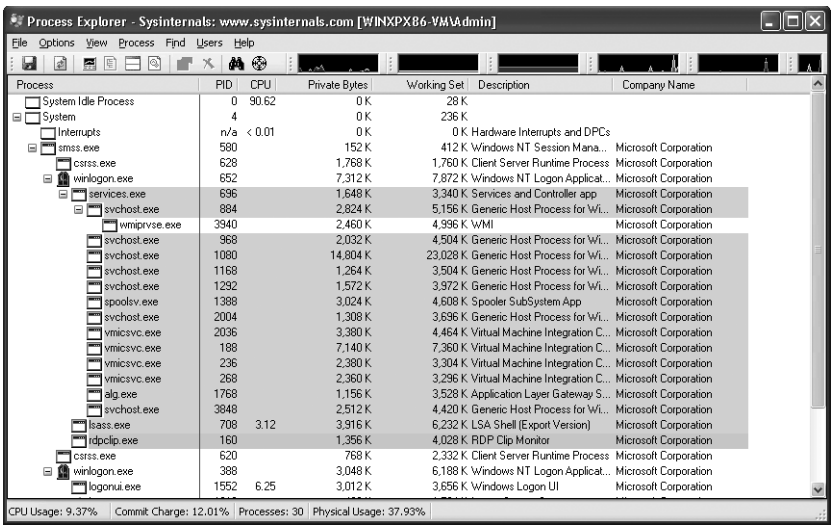


Рис. 3-4. Дерево процессов в Windows XP

В Windows Vista и выше пользователь больше не входит в один сеанс терминальных служб со службами (сеанс 0), поэтому последовательность загрузки и входа изменилась (см. рис. 3-5).

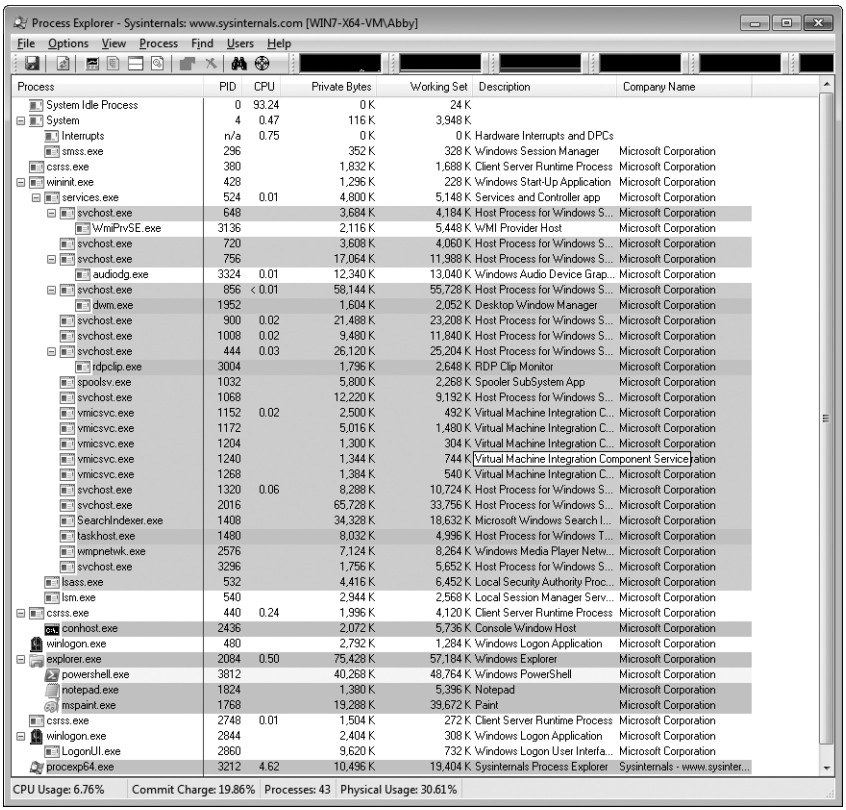


Рис. 3-5. Дерево процессов в Windows 7

Процесс System запускает экземпляр Smss.exe, который работает до выключения системы. Этот экземпляр Smss.exe запускает два новых экземпляра Smss.exe, один — в сеансе 0 и один — в сеансе 1, которые создают процессы в своих сеансах. Оба этих экземпляра завершаются перед входом пользователя, так что первоначальный экземпляр Smss.exe всегда отображается как не имеющий дочерних процессов. Экземпляр Smss.exe в сеансе 0 запускает экземпляр Csrss.exe в сеансе 0 и Wininit.exe. Wininit.exe запускает Services.exe, Lsass.exe, и Lsm.exe (Диспетчер локальных сеансов). В сеансе 1 Smss.exe запускает новый экземпляр Csrss.exe и Winlogon.exe. Winlogon запускает LogonUI.exe для запроса учетных данных интерактивного пользователя, а затем — Userinit.exe (который уже запускает Explorer после аутентификации пользователя). Как LogonUI так и Userinit обычно завершаются до инициализации оболочки, после чего пользователь может запустить Procexp. Как и в Windows XP, большинство служб являются потомками Services.exe, однако, в отличие от XP, Services.exe не содержит никаких служб. Полное дерево процессов для загрузки можно увидеть в главах, посвященных утилите Process Monitor.

Пользовательские процессы В выводе Procexp есть ряд интересных моментов и закономерностей. Так, некоторые «свои» процессы являются

дочерними процессами служб, а не Explorer. Типичные примеры — внепроцессные компоненты DCOM. Приложение вызывает компонент, и механизм COM определяет, следует ли запустить его в отдельном процессе. Этот новый процесс может работать под записью интерактивного пользователя, но запускается он процессом-контейнером службы DcomLaunch, а не пользовательской программой. Аналогично, в Windows Vista и более поздних версиях Диспетчер окон рабочего стола (Desktop Window Manager, Dwm.exe) запускается под записью пользователя Диспетчером окон рабочего стола (Desktop Window Manager, UxSms), службой Диспетчера сеансов.

Другой момент — использование объектов-заданий. Некоторые процессы-контейнеры компонентов DCOM, особенно инструментария управления Windows (WMI), работают с ограничениями по объему выделяемой им памяти, числу дочерних процессов и потребляемому времени ЦП. Все, что запускается с помощью службы вторичного входа в систему (Secondary Logon), например, командой Запуск от имени, добавляется к одному заданию, так что исходный процесс и запускаемые им дочерние процессы можно отслеживать как единое целое и завершать их после выхода пользователя из системы. Наконец, Мастер совместимости программ (Program Compatibility Assistant, PCA) в Windows Vista и выше отслеживает унаследованные приложения, предлагая решения возможных проблем с совместимостью после завершения последнего процесса в задании.

Операции с процессами

Над процессами в Procexp можно выполнять ряд операций. Для этого нужно щелкнуть процесс правой кнопкой мыши либо выделить его и выбрать нужный пункт меню **Process**:

- **(Окно)** Если у процесса имеется видимое окно на рабочем столе, то командой **Window** его можно активировать, восстановить, свернуть, развернуть или закрыть. Если у процесса нет видимых окон, то пункт **Window** деактивирован.
- **Set Affinity (Установить привязку)** В многопроцессорных системах можно привязать процесс к ЦП, т. е. сделать так, чтобы потоки этого процесса исполнялись только на указанных вами ЦП (рис. 3-6.). Это удобно, когда какой-либо процесс чрезмерно загружает ЦП, останавливать этот процесс нельзя, а нужно лишь вернуть контроль над системой для устранения неполадки. Командой **Set Affinity** можно временно запретить этому процессу работу на всех ЦП кроме одного, в результате систему станет отзываться на действия пользователя. Если требуется привязать процесс к ЦП, но нет возможности изменить его исходный код, используйте т.н. «прокладку совместимости» SingleProcAffinity. Крайнее средство — внесение сведений о привязке в заголовок PE-файла.

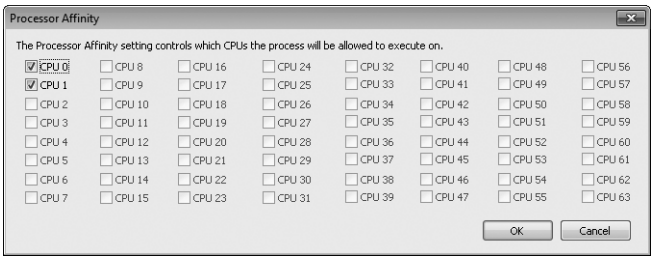


Рис. 3-6. Установка привязки к процессору на двухъядерном компьютере

- **Set Priority (Установить приоритет)** Просмотр или установка базового приоритета для процесса.
- **Kill Process (Уничтожить процесс)** Можно принудительно завершить процесс, выбрав **Kill Process** или щелкнув кнопку **Kill Process** на панели инструментов. По умолчанию Procexp запрашивает подтверждение перед завершением процесса. Этот запрос можно отключить, сняв флажок **Confirm Kill (Подтвердить уничтожение)** в меню **Options**.

Внимание! Принудительное завершение не даст процессу возможность корректно завершить работу и может привести к потере данных или нестабильности системы. Кроме того, Procexp не предупреждает о попытке завершения критических для системы процессов, например Csrss.exe. Остановка критического процесса приводит к немедленному краху Windows с «синим экраном».

- **Kill Process Tree (Уничтожить дерево процесса)** В режиме дерева этот элемент меню активируется, позволяя завершить процесс со всеми его потомками (после подтверждения, если включен параметр **Confirm Kill**).
- **Restart (Перезапуск)** Procexp останавливает выделенный процесс (после подтверждения, если необходимо) и запускает тот же образ с теми же аргументами командной строки. Имейте в виду, что новый экземпляр процесса может работать со сбоями, если он зависит от других параметров системы, таких как контекст безопасности, переменные окружения или унаследованные описатели объекта.
- **Suspend (Приостановить)** Чтобы процесс временно остановился и освободил системные ресурсы (сеть, ЦП или диск) для других процессов, можно приостановить потоки процесса. Для возобновления приостановленного процесса выберите команду **Resume (Возобновить)** из контекстного меню процесса.

Совет Приостановка полезна при обнаружении вредоносного ПО типа «buddy system» (когда два или несколько процессов отслеживают друг друга, и после принудительной остановки одного процесса остальные вновь запускают его). Для устранения подобного вредоносного ПО нужно сначала приостановить все его процессы, а затем завершить их.

- **Launch Depends (Запустить Depends)** Если обнаруживается утилита Dependency Walker (Depends.exe), Procexp запускает ее, передавая путь к исполняемому образу выбранного процесса как аргумент командной

строки. Depends.exe отображает зависимости от DLL. Данная утилита раньше поставлялась с разными программами Microsoft, а сейчас распространяется через сайт *www.DependencyWalker.com*.

- **Debug (Отладка)** Данный элемент меню доступен, только если в HKEY_LOCAL_ MACHINE\Software\Microsoft\WindowsNT\CurrentVersion\AeDebug зарегистрирован отладчик. При выборе **Debug** зарегистрированный отладчик запускается с параметром –p, за которым следует PID выбранного процесса. Имейте в виду, что закрытие отладчика без отключения от программы приводит к ее остановке. Если регистрация отладчика изменилась во время работы Procexp, то Procexp необходимо перезагрузить, чтобы изменение вступило в силу.
- **Create Dump (Создать дамп)** Запись в файл мини-дампа или полного дампа памяти выбранного процесса без завершения процесса.
- **Properties (Свойства)** Этот элемент меню отображает окно **Properties** для выбранного процесса с массой информации о нем (подробнее об этом — ниже).
- **Search Online (Искать в Интернете)** Поиск по имени выбранного исполняемого файла в браузере и поисковой системе по умолчанию. Удобен при поиске вредоносного ПО и определении программы, которой принадлежит процесс.

Настройка выбора столбцов

Можно выбрать столбцы для отображения, щелкнув правой кнопкой мыши строку заголовка столбца и выбрав пункт **Select Columns** в контекстном меню либо в меню **View**. Procexp может отобрать свыше 100 атрибутов процессов в главном окне и более 30 атрибутов в режимах просмотра DLL и описателей в строке состояния. В окне **Select Columns** (рис. 3-7) они находятся на десяти вкладках: **Process Image (Образ процесса)**, **Process Performance (Производительность процесса)**, **Process Memory (Память процесса)**, **.NET, Process I/O (Ввод-вывод процесса)**, **Process Network (Сетевая активность процесса)**, **Process Disk (Дисковая активность процесса)**, **Handle (Описатель)**, **DLL** и **Status Bar (Строка состояния)**.

Вкладка Process Image

Вкладка **Process Image** (рис. 3-7) содержит атрибуты процесса, большинство из которых устанавливается при запуске процесса и не изменяются в течение его жизненного цикла. К ним относятся столбцы **Process Name** и **PID**, которые отображаются всегда, также можно вывести другие столбцы:

- **User Name** Учетная запись пользователя, под которой работает процесс, в формате *Имя_домена\Имя_пользователя*.
- **Description (Описание)** Сведения о версии исполняемого файла из его образа. Если это столбец скрыт, информация будет отображаться в подсказке **Process Name**.

- **Company Name (Компания)** Данные из образа исполняемого файла. Если этот столбец скрыт, информация будет отображаться в подсказке **Process Name**.
- **Verified Signer (Проверенный владелец подписи)** Показывает, заверен ли образ исполняемого файла цифровой подписью с сертификатом от корневого центра сертификации, доверяемого на этому компьютере (подробнее об этом — ниже, в разделе о проверке подписей образа).
- **Version (Версия)** Данные о версии файла из информации, внедренной в образ исполняемого файла.
- **Image Path (Путь к образу)** Путь к образу исполняемого файла. Если данный столбец отображается, в подсказке **Process Name** полный путь не виден.
- **Image Type (Тип образа)** (64- или 32-разрядный) В 64-разрядных версиях Windows это поле показывает, запускает ли программа «родной» (64-разрядный) код или 32-разрядный код, работающий в WOW64 (Windows On Windows64). В 32-разрядных версиях Windows этот флажок снят.
- **Window Title (Заголовок окна)** Если у процесса есть видимые окна, содержит текст из заголовка активного окна (как на вкладке **Applications** Диспетчера задач). Этот атрибут является динамическим и изменяется при изменении заголовка окна.
- **Window Status (Состояние окна)** Если у процесса есть видимые окна, показывает, реагируют ли они на оконные сообщения (значения **Running (Работает)** или **Not Responding (Не отвечает)**). Этот столбец аналогичен столбцу **Состояние** на вкладке **Приложения** Диспетчера задач. Данный атрибут также является динамическим.
- **Session (Сеанс)** Сеанс служб терминалов, в котором работает процесс. Службы и большинство системных программ работают в сеансе 0. Пользователи в Windows XP и Windows Server 2003 могут работать в любом сеансе, а в Windows Vista и более поздних версиях пользователи всегда работают в сеансе с номером 1 или больше.
- **Command Line (Командная строка)** Командная строка, которая использовалась для запуска процесса.
- **Comment (Комментарий)** Комментарий, который пользователь может задать на вкладке **Image** в окне **Properties** (подробнее см. ниже).
- **DEP Status (Состояние DEP)** Показывает, включена ли защита от исполнения данных (DEP) для процесса. DEP — это функция защиты от атак переполнением буфера и других атак, функционирует путем запрета исполнения содержимого областей памяти, помеченных флагом «no-execute» («неисполняемая»), таких как стек или куча. Это поле может быть пустым (DEP отключена) либо содержать текст *DEP* (enabled) (DEP включена), *DEP (permanent)* (DEP включена в исполняемом файле и не

может быть отключена) или *<n/a>* , если Procexp не может определить состояние DEP для процесса.

- **Integrity Level (Уровень целостности)** В Windows Vista и выше показывает уровень целостности (IL) процесса. Службы работают на системном уровне (System), процессы с повышенными привилегиями — на высоком (High), стандартные пользовательские процессы — на среднем (Medium), а процессы с низким уровнем прав, например Internet Explorer в безопасном режиме — на низком (Low) IL.
- **Virtualized (Виртуализация)** В Windows Vista и выше показывает, включена ли виртуализация файла UAC и реестра. Виртуализация файлов и реестра — это технология обеспечения совместимости, которая при попытке унаследованных процессов среднего IL изменить защищенные области системы незаметно для процесса перенаправляет их в области, которые разрешено модифицировать пользователю.
- **ASLR Enabled (Включена ASLR)** В Windows Vista и выше показывает, включена ли функция ASLR (рандомизация адресного пространства) для процесса. ASLR — это средство эшелонированной защиты, препятствующее удаленным атакам через входные точки функций, расположенные по стандартным адресам в памяти.

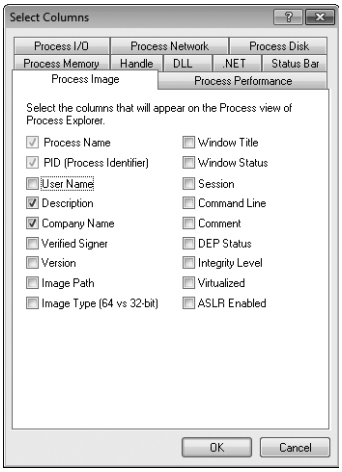


Рис. 3-7. Вкладка Process Image окна Select Columns

Для заполнения большинства вышеперечисленных полей данными из процессов, работающих в другом контексте защиты, Procexp требует прав администратора. Исключениями являются текст заголовка и состояние для окон, открытых на том же рабочем столе, что и Procexp, а также **User Name** при работе в Windows XP. Поскольку отображение атрибута **Comment** зависит от пути к образу, отображаемые данные зависят от того, был ли комментарий введен во время работы Procexp с тем же уровнем прав, что в данный момент.

Вкладка Process Performance

Вкладка **Process Performance** (рис. 3-8) содержит данные о загруженности ЦП, числе потоков и открытых описателей у процесса. Одни атрибуты сообщают сводные данные, другие — изменения с момента предыдущего обновления. Procexp не требует прав администратора для получения информации для данной вкладки (подробнее об оценке производительности в Procexp см. выше).

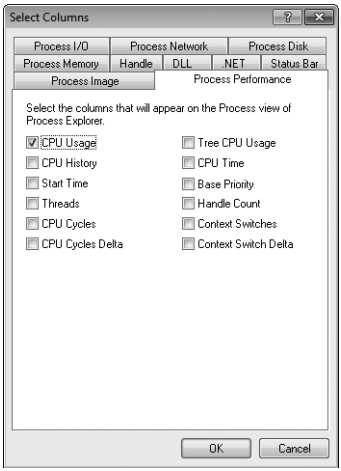


Рис. 3-8. Вкладка Process Performance в окне Select Columns

Все перечисленные ниже столбцы, за исключением **Start Time**, содержат динамические атрибуты, которые обновляются при каждой регенерации.

- **CPU Usage (Загруженность ЦП)** Время, использованное процессом (или псевдопроцессом) с момента предыдущего обновления (в процентах от общего времени ЦП, с округлением до двух десятичных знаков). В Windows 7 это поле содержит < 0.01, если процесс использовал менее сотой доли процента общего времени ЦП либо пусто, если процесс вовсе не использовал время ЦП в данный промежуток времени. В Windows Vista и более ранних версиях это поле содержит < 0.01, только если поля C**PU Cycles Delta** и **Context Switch Delta** не скрыты и отображают прирост использованного времени ЦП и числа переключений контекста за данный период. Пустые поля означают, что процесс не исполнялся или использовал менее 0,01% времени ЦП. Значение можно округлять до целых, а не до сотых с помощью параметра **Show Fractional CPU** в меню **View** (подробнее об измерении загруженности ЦП см. выше).
- **Tree CPU Usage (Дерево загруженности ЦП)** Проценты времени ЦП, использованного процессом и всеми его потомками. Имейте в виду, что значения **Tree CPU Usage** рассчитываются только на основе таймера (подробнее об измерении загруженности ЦП см. выше).
- **CPU History (История ЦП)** Графическое представление текущей загруженности ЦП каждым из процессов. Время ядра выделяется красным цветом, а время пользователя — зеленым.

- **CPU Time (Время ЦП)** Общее количество времени ЦП в режиме ядра и пользователя, затраченное процессом (или псевдопроцессом) в формате ЧЧ:ММ:СС.МС.
- **Start Time (Время запуска)** Время и дата запуска процесса.
- **Base Priority (Базовый приоритет)** Приоритет планирования процесса: 8 —обычный приоритет, больше 8 — повышенный, меньше 8 — пониженный приоритет.
- **Threads (Потоки)** Число потоков в процессе.
- **Handle Count (Счетчик описателей)** Число описателей объектов ядра, открытых процессом в данный момент.
- **CPU Cycles (Циклы ЦП)** В Windows Vista и выше — общее время ЦП в режиме ядра и пользовательском режиме, использованное процессом с момента запуска (В Windows Vista это число не отслеживается для псевдопроцесса **Interrupts**).
- **CPU Cycles Delta (Прирост времени ЦП)** В Windows Vista и выше — время ЦП, затраченное процессом с момента предыдущего обновления (В Windows Vista это число не отслеживается для псевдопроцесса **Interrupts**).
- **Context Switches (Переключения контекста)** Общее число переключений контекста ЦП для начала исполнения потока (для псевдопроцесса **Interrupts** — число DPC и прерываний). Windows не подсчитывает переключения контекста по процессам, поэтому данный атрибут показывает число переключений контекст для всех потоков. Если поток завершается, соответствующие переключения контекста не добавляются в это значение.
- **Context Switch Delta (Прирост переключений контекста)** Число переключений контекста ЦП с момента последнего обновления (для псевдопроцесса **Interrupts** — число DPC и прерываний с момента последнего обновления).

Вкладка Process Memory

Вкладка **Process Memory** (рис. 3-9) содержит атрибуты, связанные с использованием памяти, включая виртуальную память, рабочий набор и страничные ошибки, а также число подсчеты объектов оконной системы (GDI и USER).

Все эти свойства — динамические, обновляемые при каждой регенерации. Большинство этих параметров удается прочитать у любых процессов без прав администратора. В случае процессов, работающих в других контекстах защиты, Procexp требует прав администратора для чтения следующих параметров: минимальный и максимальный размеры рабочего наборы, общий рабочий набор, разделяемая и закрытая память, а также счетчики объектов GDI и USER. Кроме того, независимо от привилегий, счетчики GDI и USER можно читать только для процессов в том же сеансе служб терминалов.

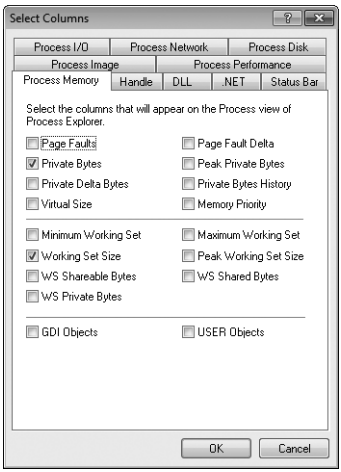


Рис. 3-9. Вкладка Process Memory в окне Select Columns

- **Page Faults (Страничные ошибки)** Общее число обращений процесса к недействительным страницам памяти, приведших к вызову обработчика ошибок Диспетчера памяти. Недействительной страница может быть по ряду причин: страница сброшена на диск в страничный или проецируемый файл, страница должна быть скопирована или обнулена при первом обращении либо имел место несанкционированный доступ, приведший к нарушению прав доступа. Учтите, что в это число входят и «мягкие» страничные ошибки (то есть, ошибки, устраненные путем обращения к нужным данным, уже загруженным в оперативную память, а не просто данным из рабочего набора).
- **Page Fault Delta (Прирост страничных ошибок)** Число страничных ошибок, возникших с момента последнего обновления окна программы. Обратите внимание: заголовок данного столбца — «PF Delta».
- **Private Bytes (Байты закрытой памяти)** Количество выделенных и переданных байтов памяти для единоличного (а не совместного с другими процессами) использования процессом. Закрытая память включает кучу и стек. Постоянный рост данного показателя может свидетельствовать об утечке памяти.
- **Private Delta Bytes (Изменение закрытой памяти)** Число байтов, на которое увеличилась или уменьшилась закрытая память процесса с момента предыдущего обновления.
- **Peak Private Bytes (Пиковый размер закрытой памяти)** Максимальный размер закрытой памяти (в байтах) единовременно переданной процессу с момента его запуска.
- **Private Bytes History (Выделение закрытой памяти)** График передачи закрытой памяти процессу. Чем шире этот столбец, тем дольше отображаемый временной интервал. Обратите внимание: график масштабируется по процессу с максимальным на текущий момент размером закрытой памяти.

- **Virtual Size (Размер виртуальной памяти)** Объем зарезервированной или переданной процессу виртуальной памяти.
- **Memory Priority (Приоритет памяти)** (в Windows Vista и выше) Приоритет по умолчанию, присвоенный страницам физической памяти, используемым процессом. Кешированные в ОЗУ и не включенные ни в один рабочий набор страницы с низким приоритетом первыми передаются для повторного использования.
- **Minimum Working Set (Минимальный размер рабочего набора)** Минимальный размер физической памяти, зарезервированный для процесса, который гарантированно выделяется под его рабочий набор операционной системой. Процессу разрешено блокировать страницы рабочего набора в количестве, равном данному значению за вычетом восьми страниц. Минимальный размер рабочего набора может и не соблюдаться, если приложение, управляющее ресурсами, не устанавливает жесткие ограничения.
- **Maximum Working Set (Максимальный размер рабочего набора)** Наибольший размер рабочего набора процесса. Впрочем, Windows игнорирует этот параметр, если приложение, управляющее ресурсами, не установит для процесса жесткие ограничения.
- **Working Set Size (Размер рабочего набора)** Объем физической памяти, переданной процессу Диспетчером памяти.
- **Peak Working Set Size (Пиковый размер рабочего набора)** Максимальный размер рабочего набора процесса с момента его запуска.
- **WS Shareable Bytes (Разделяемая часть рабочего набора)** Память в составе рабочего набора процесса, которую процесс может использовать совместно с другими процессами, например, проекции исполняемых образов.
- **WS Shared Bytes (Общая часть рабочего набора)** Память в составе рабочего набора процесса, которая в данный момент используются совместно с другими процессами.
- **WS Private Bytes (Закрытая часть рабочего набора)** Память в составе рабочего набора процесса, которая не может использоваться совместно с другими процессами.
- **GDI Objects (Объекты GDI)** Число принадлежащих процессу объектов интерфейса графических устройств (GDI): кистей, шрифтов и растровых изображений.
- **USER Objects (Объекты USER)** Число объектов USER, в частности, окон и меню, принадлежащих процессу.

Имейте в виду, что объекты GDI и USER создаются оконной подсистемой в сеансе сервера терминалов, в котором работает процесс. Они не являются объектами ядра и не имеют связанных с ними дескрипторов защиты.

Вкладка .NET

Вкладка .NET (рис. 3-10) содержит различные счетчики для процессов, использующих .NET framework версии 1.1 и выше.

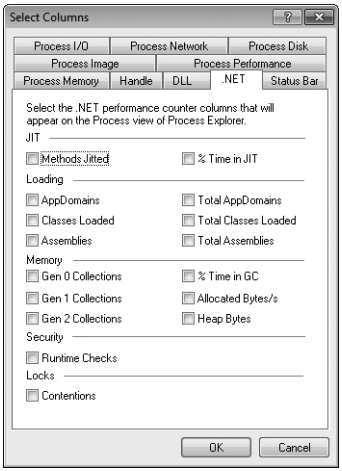


Рис. 3-10. Вкладка .NET в окне Select Columns

Все эти значения являются динамическими. Для просмотра этих данных у процессов, работающих в другом контексте защиты, требуются права администратора.

- **Methods Jitted (JIT-компилированные методы)** Общее число методов, скомпилированных по запросу с момента запуска приложения.
- **% Time in JIT (Доля времени на JIT-компиляцию)** Процент времени, затраченного на JIT-компиляцию, с момента последней фазы JIT-компиляции.
- **AppDomains (Домены приложения)** Число доменов приложения, загруженных в данный момент приложением.
- **Total AppDomains (Всего доменов приложения)** Максимальное число доменов приложения, загруженных с момента запуска приложения.
- **Classes Loaded (Загруженные классы)** Число классов во всех сборках, загруженных в данный момент.
- **Total Classes Loaded (Всего загруженных классов)** Совокупное число классов, загруженных во всех сборках с момента запуска приложения.
- **Assemblies (Сборки)** Число сборок, загруженных в данный момент во все домены приложения в работающем приложении. Постоянный рост данного значения может свидетельствовать об утечке памяти, занятой сборками.
- **Total Assemblies (Всего сборок)** Отображает общее число сборок, загруженных с момента запуска приложения.
- **Gen 0, 1, 2 Collections (Уничтожение объектов поколений 0, 1, 2)**

Сколько раз выполнялся сбор мусора среди объектов поколений нулевого, первого и второго поколений с момента запуска приложения. Объекты поколения 0 — самые новые, недавно созданные, а объекты поколения 2 полностью освобождаются при сборе мусора (называемом в данном случае *полным сбором мусора*). Число проходов сбора мусора среди объектов младшего поколения включает проходы среди объектов всех старших поколений.

- **% Time in GC (Доля времени, затраченного на сбор мусора)** Время (в процентах), затраченное на сбор мусора с момента последнего цикла сбора мусора.
- **Allocated Bytes/s (Выделение памяти, байтов/с)** Память (в байтах в секунду), выделенных в куче после сбора мусора.
- **Heap Bytes (Байты кучи)** Число байтов, выделенных во всех кучах процесса.
- **Runtime Checks (Проверки исполняющей среды)** Общее число проверок прав при обращении к коду исполняющей среды с момента запуска приложения.
- **Contentions (Конфликты)** Общее число безуспешных попыток получения потоками управляемых блокировок во время выполнения.

Вкладка Process I/O

Вкладка **Process I/O** (рис. 3-11) содержит сведения о файловом вводе-выводе и вводе-выводе на устройствах, в том числе через редиректоры LANMan и WebDAV. При отображении этих столбцов Procexp подсчитывает системные вызовы NtReadFile, NtWriteFile и NtDeviceloControlFile, представляющие операции чтения, записи и другие операции ввода-вывода (соответственно, категории Read, Write, Other), а также число байтов, переданных и полученных при этих вызовах. Счетчики, отображаемые Procexp, предназначены для «частного» ввода-вывода, то есть, однозначно связанных с процессом. Имейте в виду, что ввод-вывод проекции файла не всегда удается связать с одним процессом.

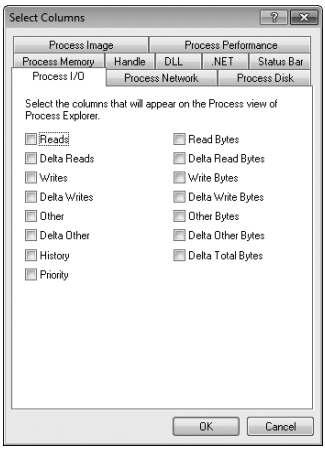


Рис. 3-11. Вкладка Process I/O в окне Select Columns

Все они являются динамическими и обновляются вместе с окном программы. Для чтения соответствующих данных о процессах, работающих под другой учетной записью, необходимы права администратора, чего не требуется для процессов, работающих под одной учетной записью с Procexp, даже на более высоком IL.

По умолчанию Procexp сообщает точное число байтов. Если включить параметр **I/O Bytes Columns (Единицы ввода-вывода)** в меню **View**, Procexp будет показывать приближенные значения в Кб, Мб или Гб. Заметьте, что отображаемые имена атрибутов в заголовках столбцов начинаются с «I/O». Например, если включить отображение счетчика **Read Bytes (Прочитанные байты)** на данной вкладке, то заголовок соответствующего столбца будет «I/O Read Bytes».

- **I/O operations (Операции ввода-вывода)** Существует четыре счетчика для таких операций ввода-вывода как чтение, записи и другие: общее количество операций, выполненных процессом с момента его запуска (например, Reads), общее число прочитанных или записанных байтов (например, Read Bytes), число операций, выполненных с момента последнего обновления окна Procexp (Delta Reads), и число байтов, прочитанных (Delta Read Bytes) или записанных с момента последнего обновления.
- **Delta Total Bytes (Суммарное изменение, байтов)** Суммарное число байтов, прочитанных или записанных в операциях ввода-вывода с момента предыдущего обновления.
- **I/O History (Журнал ввода-вывода)** График количества информации, прошедшей через процесс. Синяя линия обозначает общий трафик, а розовая — динамику объема записанных данных.
- **I/O Priority (Приоритет ввода-вывода)** Приоритет ввода-вывода для процесса (только в Windows Vista и выше). Назначение приоритетов ввода-вывода позволяет подсистемам ввода-вывода различать активные и фоновые процессы с низким приоритетом. Большинству процессов назначен приоритет Normal, в то время как у остальных он может быть Low (Низкий) или Very Low (Очень низкий). Только Диспетчеру памяти назначен приоритет ввода-вывода Critical (Критический). Приоритет пятого уровня — High (Высокий) — не используется в текущих версиях Windows.

Вкладка Process Network

Вкладка **Process Network** (рис. 3-12) позволяет увидеть в Procexp число операций установления и разрыва соединений, отправки и получения данных по протоколу TCP, число байт, принятых и отправленных в данных операциях, а также изменения этих значений с момента предыдущего обновления. Имейте в виду, что эти цифры не охватывают файловый ввод-вывод через редиректор LANMan но включают файловый ввод-вывод через редиректор WebDAV.

Учтите также, что для вывода всех атрибутов на данной вкладке нужны права администратора. Если Procexp работает без администраторских прав, вкладка Process Network не отображается в окне Select Columns. Procexp выводит предупреждение при включении отображения любого из этих столбцов и запуске Procexp без прав администратора.

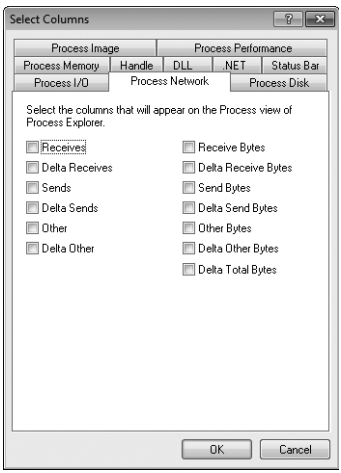


Рис. 3-12. Вкладка Process Network в окне Select Columns

Как и на вкладке **Process I/O**, на **Network I/O** отображается общее число операций **Receives (Получение)**, **Sends (Отправка)** и **Other (Другие)** с момента запуска процесса и предыдущего обновления, а также число байтов с момента запуска процесса и с момента предыдущего обновления.

Суммарные показатели, отображаемые Procexp при включении вывода этих столбцов, отражают только общее число операций, а также принятых и отправленных байтов с момента запуска Procexp. Windows не отслеживает эти показатели для отдельных процессов, поэтому реконструировать динамику сетевой активности до запуска Procexp невозможно.

По умолчанию Procexp выводит точное число байтов. Если активировать параметр **I/O Bytes Columns** в меню **View**, будут выводиться приближенные значения в кило-, мега- или гигабайтах.

Вкладка Process Disk

На вкладке **Process Disk** (рис. 3-13) включается отображение активности ввода-вывода для локальных дисков (за исключением оптических приводов). В отличие от вкладки **Process I/O**, здесь содержатся все операции ввода-вывода для диска, в том числе инициированные компонентами ядра и драйверами файловой системы. Сюда не входят операции файлового ввода-вывода для сетевых редиректоров или кеш-буферов в памяти.

Обратите внимание: отображение любого атрибута с данной вкладки требует прав администратора. Если Procexp работает без прав администратора, вкладка **Process Disk** в окне **Select Columns** не отображается. Procexp вы-

водит предупреждение при включении вывода любого из этих атрибутов и запуске Procexp без прав администратора.

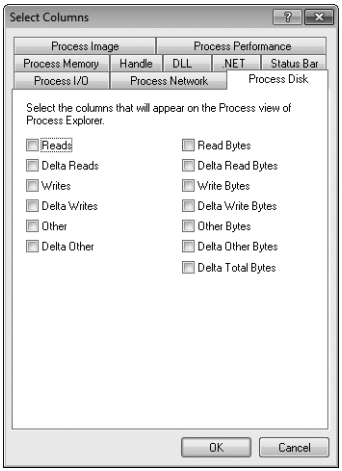


Рис. 3-13. Вкладка Process Disk в окне Select Columns

Подобно счетчикам на вкладках **Process I/O** и **Process Network**, таковые на вкладке **Disk I/O** выводят общее число операций (чтения, записи и прочих) и байтов, прочитанных или записанных с момента запуска процесса и с момента предыдущего обновления. Как и на вкладке **Network I/O**, в полях **Process Disk** суммарные показатели отражают только число операций и байтов с момента запуска Procexp, а все, что было до него, Procexp просто не «видит».

По умолчанию Procexp выводит точное число байтов. Если активировать параметр **I/O Bytes Columns** в меню **View**, Procexp будет отображать приближенные значения в Кб, Мб или Гб.

Сохранение настроек столбцов

Можно сохранить конфигурацию столбца, включая порядок сортировки, выбрав пункт **Save Column Set (Сохранить настройки столбцов)** в меню **View**. Procexp попросит дать имя настройкам (рис. 3-14). Для изменения существующих настроек столбца сохраните измененную конфигурацию под тем же именем, что и настройки, которые требовалось изменить, выбрав ее в раскрывающемся списке.

Чтобы загрузить ранее сохраненные настройки столбцов, выберите их в меню **Load Column Set (Загрузить настройки столбцов)** в меню **View** или нажмите заданную комбинацию клавиш (она отображается в меню). Для того чтобы переименовать, изменить или удалить существующие настройки, выберите пункт **Organize Column Sets (Управление настройками столбцов)** в меню **View**. Если изменить порядок сортировки, изменится порядок отображения настроек в меню **Load Column Set** и назначенные им клавиатурные комбинации.

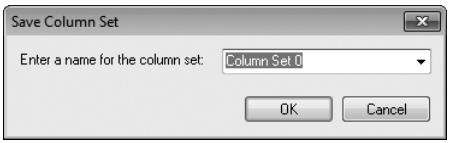


Рис. 3-14. Окно Save Column Set

Примечание Клавиатурные комбинации для доступа к сохраненным настройкам столбцов в Procexp конфликтуют с клавиатурными комбинациями утилиты Zoomit, подробнее см. в главе 10.

Сохранение отображаемых данных

Щелчок кнопки **Save** на панели инструментов сохраняет «моментальный снимок» активности текущего процесса в текстовый файл. Procexp сохраняет данные всех столбцов, выбранных для отображения в главном окне, а также в нижней панели (если она открыта) в текстовый файл с разделителями-табуляторами. Если имя и путь файла для сохранения еще не выбраны, Procexp попросит указать их. Чтобы задать новое имя и путь, выберите пункт **Save As** в меню **File**.

Панель инструментов

Панель инструментов Procexp содержит кнопки для быстрого доступа к часто используемым функциям, а также четыре или шесть постоянно обновляющихся графиков ряда параметров системы.

Рис. 3-15. Меню и панель инструментов Procexp

Графики

Допускается изменять размер мини-графиков на панели инструментов Procexp и выносить их в отдельные строки путем перетаскивания их мышью за вертикальный маркер у левого края (рис. 3-15). Procexp отображает графики загруженности ЦП, выделения памяти, утилизации физической памяти, файлового ввода-вывода и ввода-вывода на устройствах. Если Procexp работает с правами администратора, добавляются графики сетевого и дискового ввода-вывода.

График ЦП показывает динамику изменения общей загруженности ЦП, красным отображается загруженность в режиме ядра, а зеленым — общая загруженность в режимах ядра и пользователя. Суммарная динамика выделения памяти в системе показана желтым цветом, а загруженность физической памяти — оранжевым. Текущая интенсивность ввода-вывода системы в целом отображается фиолетовым цветом (операции записи) и голубым (суммарные показатели ввода-вывода). Если навести указатель мыши на графики, выводится всплывающая подсказка, содержащая значения в точке под указателем и соответствующее ему время. Для графиков загружен-

ности ЦП и активности ввода-вывода отображается процесс, наиболее интенсивно загружающий ЦП и выполняющий ввод-вывод в данный момент. Чем шире график, тем более длительный промежуток времени он охватывает. Если щелкнуть любой график, он откроется в окне **System Information (Информация о системе)**, подробнее о графиках — ниже.

Можно вывести уменьшенные версии этих графиков (и соответствующих подсказок) в области уведомлений панели задач (на жаргоне — «системный трей»), выбрав соответствующий параметр в меню **Options | Tray Icons (Параметры | Значки области уведомлений)**. По умолчанию отображается только значок **CPU Usage** с динамикой загруженности ЦП за короткий предшествующий период. Загруженность в режиме ядра обозначается красным, суммарная загруженность — зеленым. Если щелкнуть любой значок Procexp в области уведомлений, откроется главное окно утилиты.

Если щелкнуть правой кнопкой значок Procexp в области уведомлений, откроется контекстное меню, из которого можно вызвать окно **System Information** или главное окно, а также выйти из Procexp. Пункт **Shutdown (Выключение)** позволяет выйти из системы, заблокировать систему, выключить компьютер или перезагрузить Windows.

Кнопки панели инструментов

Ниже приводится перечень кнопок панели инструментов Procexp и разделов данной главы, где описаны их функции.

Рис. 3-16. Кнопки панели инструментов Procexp

Как показано на рис. 3-16, значки панели инструментов располагаются в следующем порядке:

- **Save** — «Сохранение отображаемых данных».
- **Refresh Now** — «Обновление окна».
- **System Information** — «Информация о системе».
- **Show Process Tree** — «Дерево процессов».
- **Show/Hide Lower Pane** — «Динамически подключаемые библиотеки и описатели».
- **View DLLs / View Handles** — «Режим просмотра DLL» и «Режим просмотра описателей».
- **Properties** — открывает окно **Properties** для выбранного процесса, описателя или DLL.
- **Kill Process/Close Handle** — завершает выбранный процесс и закрывает описатель, если он выбран в режиме просмотра описателей (как сказано выше, эти операции, особенно закрытие используемого процессом описателя, довольно рискованны).
- **Find Handle or DLL** — «Поиск DLL и описателей».
- **Find Window's Process** — «Поиск процесса — владельца окна».

Поиск процесса — владельца окна

С помощью Procexp можно быстро найти процесс, которому принадлежит любое окно на рабочем столе. Перетащите значок инструмента в виде перекрестья прицела с панели инструментов на интересующее вас окно. При этом окно Procexp может скрыться, а нужное окно будет подсвечено рамкой. Отпустите кнопку мыши — откроется окно Procexp с процессом, владеющим выбранным окном. Эта функция особенно удобна для поиска источника сообщений об ошибках.

Строка состояния

В строке состояния отображаются значения ключевых параметров системы: загруженности ЦП, число процессов и использование памяти. Если автоматическое обновление Procexp отключено, в строке состояния выводится слово «Paused» («Приостановлено»).

Если щелкнуть строку состояния правой кнопкой и выбрать **Select Status Bar Columns (Выбрать поля строки состояния)**, можно настроить отображаемые параметры (рис. 3-17). Доступны некоторые общесистемные параметры и параметры отдельных процессов, работающих под одной учетной записью с Procexp. Команда **Refresh Time (Время обновления)** отображает время последнего обновления Procexp.

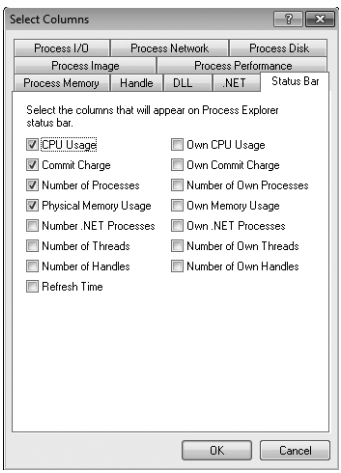


Рис. 3-17. Вкладка Status Bar в окне Select Columns

Динамически подключаемые библиотеки и описатели

Нижняя панель Procexp позволяет заглянуть внутрь процесса, выбранного в верхней панели, и увидеть его содержимое. В режиме просмотра DLL отображаются все библиотеки и другие файлы, отображаемые в адресное про-

странство процесса, а в режиме просмотра описателей видны все объекты ядра, открытые процессом. Режим просмотра DLL (рис. 3-18) активируется нажатием Ctrl + D, а режим просмотра описателей — нажатием Ctrl+H; нажатия Ctrl+L открывают и закрывают нижнюю панель. Для изменения относительных размеров панелей нужно перетащить разделитель панелей.

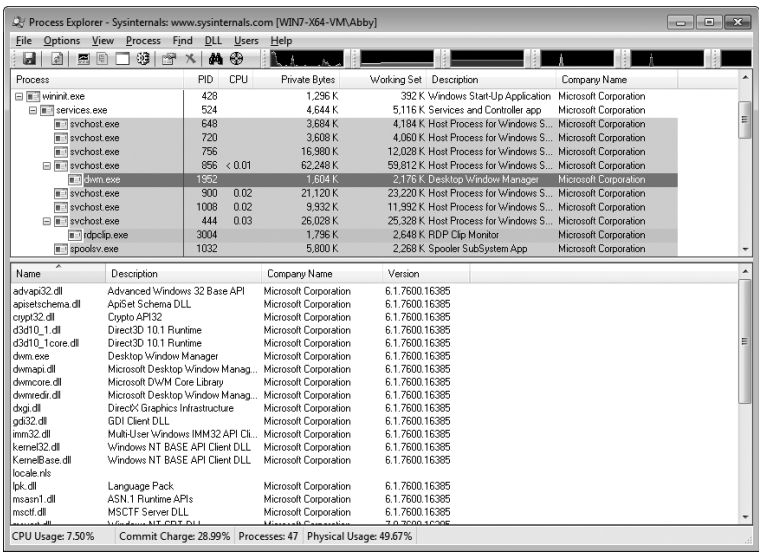


Рис. 3-18. Нижняя панель Procexp в режиме просмотра DLL

Поиск DLL и описателей

К наиболее удобным функциям Procexp относится возможность быстрого поиска процессов, в которые загружена некоторая DLL или открыт определенный объект. Это полезно, например, когда при попытке удалить какую-нибудь папку Windows не дает сделать это, сообщая, что папка «открыта другой программой», но не называет эту программу.

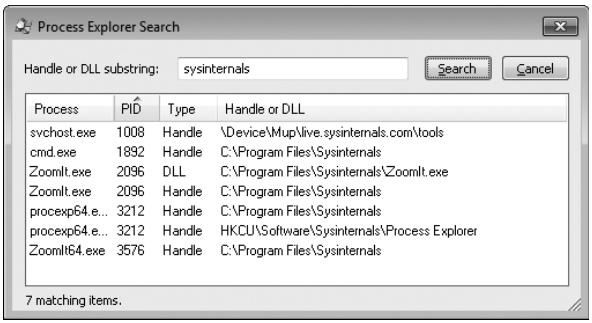


Рис. 3-19. Окно Search Process Explorer

Нажмите Ctrl + F, чтобы открыть окно **Search** (рис. 3-19), введите имя или часть имени DLL либо объекта, который вы хотите найти, а затем щел-

кните кнопку **Search.** Procexp сверит введенное вами имя со списком DLL, типом и именем доступных описателей, а затем покажет список процессов и принадлежащих им DLL и объектов, соответствующих критерию поиска. Щелкните результат, чтобы открыть его на верхней и нижней панелях. При двойном щелчке результаты выделяются, а окно **Search** закрывается.

Если результатов найдено очень много, щелкните заголовок столбца, чтобы отсортировать его — так легче найти нужные объекты. Столбец **Type (Тип)** определяет, чем является найденный объект — DLL (точнее, проецируемым файлом) или описателем объекта. Столбец **Handle** или **DLL** содержит имя описателя или путь к DLL. Имя описателя может отсутствовать, если в меню **View** активен параметр **Show Unnamed Handles And Mappings (Отображать безымянные описатели и проекции)**, а искомое имя совпадает с типом описателя.

Режим просмотра DLL

Как ясно из его названия, в режиме просмотра DLL отображаются все DLL, загруженные выбранным процессом. В нем отображаются и другие проецируемые файлы, в том числе файлы данных, а также файл исполняемого образа (EXE-файл). Применительно к процессу **System**, в режиме просмотра DLL отображаются образы, спроецированные в память ядра, в том числе, ntoskrnl.exe и все загруженные драйверы устройств. Для процесса **System Idle** и псевдопроцессов **Interrupts** в режиме просмотра DLL не отображается ничего.

Procexp требует прав администратора для просмотра DLL, загруженных в процессы, работающие под другой учетной записью, для просмотра образов, загруженных в процесс **System**, администраторских прав не требуется.

Настройка режима просмотра DLL

Откройте режим просмотра DLL, щелкните правой кнопкой заголовок столбца на нижней панели и выберите **Select Columns** — откроется окно **Select Columns** с вкладкой **DLL** (рис. 3-20). На этой вкладке можно выбрать атрибуты DLL и проецируемых файлов, отображаемые в режиме просмотра DLL:

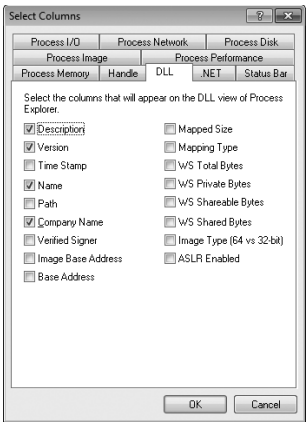


Рис. 3-20. Вкладка DLL в окне Select Columns

- **Description** Информация из внедренного в файла ресурса (если имеется).
- **Version** Информация о версии из внедренного в файла ресурса (если имеется).
- **Time Stamp** Время последнего изменения файла по данным файловой системы.
- **Name** Имя DLL или проецируемого файла либо — для безымянной проекции файла —строка <Pagefile Backed> (объект, сохраненных в страничном файле). Наведите указатель мыши на имя файла — откроется всплывающая подсказка, содержащая его полный путь.
- **Path** Полный путь к DLL или проецируемому файлу либо строка <Pagefile Backed> для безымянного проецируемого файла.
- **Company Name** Информация из внедренного в файла ресурса (если имеется).
- **Verified Signer** Показывает, был ли образ исполняемого файла заверен цифровой подписью с сертификатом от корневого центра сертификации, доверяемого на этом компьютере (подробнее о подписании образов см. ниже).
- **Image Base Address** Базовый адрес. Для файлов, загруженных как исполняемые образы, это адрес виртуальной памяти из заголовка исполняемого образа, который показывает, по какому адресу следует загрузить образ. Если этот адрес уже занят, образ загружается по другому адресу.
- **Base Address** Базовый адрес (адрес виртуальной памяти), по которому файл фактически загружен.
- **Mapped Size** Размер непрерывной области (в байтах), начиная с базового адреса, занятой проекцией файла.
- **Mapping Type** Тип проекции: «image» для исполняемых файлов-образов и «Data» для файлов данных, включая DLL, содержащие только ресурсы (например, значки и локализованный текст) и безымянные проекции файлов.
- **WS Total Bytes** Общий размер рабочего набора (байтов физической памяти, занятой в настоящее время проекциями файлов).
- **WS Private Bytes** Объем физической памяти, занятой проецируемыми файлами, и преданной данному процессу для монопольного (а не совместного с другими процессами) использования.
- **WS Shareable Bytes** Объем физической памяти, занятой проецируемыми файлами, для которой разрешено совместное с другими процессами использование.
- **WS Shared Bytes** Объем физической памяти, занятой файлами, спроецированными в адресное пространство нескольких процессов.
- **Image Type (64 vs 32-bit)** (только в 64-разрядных версиях Windows) Тип образа, указанный в заголовке файла: 64- или 32-разрядный.

- **ASLR Enabled** (только в Windows Vista и выше) В заголовке образа файла указано, что он поддерживает рандомизацию адресного пространства. Если образ не поддерживает ASLR, то столбец остается пустым, для файлов данных отображается строка «n/a».

Хотя DLL, загруженные по адресу, отличному от заданного для них базового адреса, по умолчанию не подсвечиваются, их можно включить, выбрав пункт **Relocated DLLs (Перемещенные DLL)** в окне **Configure Highlighting** (см. выше). DLL, которые не удается загрузить по стандартному для них базовому адресу из-за того, что эта область уже занята (туда спроецированы другие файлы), загрузчик перемещает в другие области памяти. Этот процесс дополнительно загружает ЦП, к тому же совместное использование перемещенных DLL становится невозможным, что снижает эффективность управления памятью Windows.

Если в меню **View** выбран пункт **Show Unnamed Handles And Mappings**, то в режиме просмотра DLL отображаются и безымянные проекции файлов, помеченные в столбцах **Name** и **Path** (если те не скрыты) как <Pagefile Backed>. Для безымянных проекций многие поля, в том числе отображаемые по умолчанию, не содержат полезной информации. Сведения о безымянных проекциях содержатся в столбцах **Base address**, **Mapped size** и **Working set metrics**.

В режиме просмотра DLL меню DLL содержит следующие пункты для именованных файлов:

- **Properties** Открывает окно **Properties** для выбранного файла. Подробнее об этом — следующем разделе.
- **Search Online** Procexp запускает поиск в Интернете по имени выбранного файла, используя браузер и поисковую систему по умолчанию. Это удобно при поиске вредоносного ПО или идентификации неизвестной DLL.
- **Launch Depends** Если установлена утилита Dependency Walker (Depends.exe, о ней см. выше), Procexp запускает ее, передавая путь к выбранному файлу как аргумент командной строки. Depends. exe показывает DLL, зависимые от данной библиотеки, и DLL, которые зависят от нее.

Анализ DLL

Дважды щелкните именованный элемент в режиме просмотра DLL, чтобы отобразить для него окно **Properties** (рис. 3-21). Вкладка **Image** отображает о проецируемом файле следующую информацию: описание, название компании, версию, путь, базовый адрес и размер в памяти процесса, а также (в 64-системах) тип (32- или 64-разрядный). Можно выделить эти поля и скопировать в буфер обмена их содержимое.

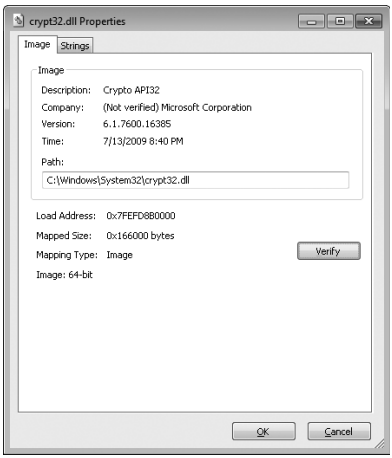


Рис. 3-21. Вкладка DLL в окне Select Columns

Поле **Company** также используется для проверки цифровых подписей исполняемых файлов, подписанных их производителями (подробнее об этом — ниже, в разделе о проверке подписи образов). Если в проецируемый файл внедрен ресурс с названием компании и номером версии, а подпись еще не проверена, щелкните кнопку **Verify**. Эта функция полезна, когда нужно убедиться, что файл действительно получен от своего производителя и не был изменен посторонними. Если подпись проходит проверку, в поле **Company** отображается строка **(Verified)** и имя владельца сертификата, с которым подписан файл. Если проверка не выполнена, в поле отображается строка **(Not verified)** и название компании, взятое из файла образа. Если образ не подписан или проверка подписи не прошла, в столбце отображается строка **(Unable to verify)** и название компании.

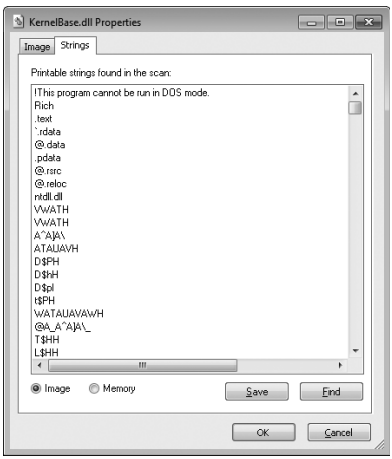


Рис. 3-22. Вкладка Strings диалогового окна DLL Properties

Вкладка **Strings (Строки)** окна **Properties** (рис. 3-22) отображает все последовательности из трех и более пригодных для печати символов, найден-

ных в проецируемом файле. Если установлен переключатель **Image**, строки читаются из дискового файла образа, а если установлен переключатель **Memory**, то из проекции файла в памяти. Строки, прочитанные из дискового файла и памяти, могут различаться, если на диске образ хранится в сжатом или зашифрованном виде и распаковывается либо расшифровываются при загрузке в память. Строки из памяти могут также включать динамически создаваемые области данных раздела памяти образа.

Примечание В программировании термином «строка» означают структуру данных из последовательности знаков, как правило, образующих понятный текст.

Щелкните кнопку **Save,** чтобы сохранить отображаемые строки в текстовый файл. Чтобы сравнить строки из файла и из памяти, сохраните их в разные отдельные файлы и откройте в программе для сравнения текстов.

Для поиска строки в списке щелкните кнопку **Find** — откроется стандартное окно поиска. Для поиска других вхождений строки просто нажмите F3 или снова щелкните **Find** либо **Find Next** — поиск продолжится с текущей выбранной строки.

Режим просмотра описателей

В режиме просмотра описателей Procexp отображает описатели, принадлежащие выбранному на верхней панели процессу (рис. 3-23). Описатели программы используют для операций с системными объектами, которыми управляет код режима ядра. К ним относятся файлы, разделы реестра, синхронизирующие объекты, разделы памяти, оконные станции и рабочие столы. Несмотря на разнообразие ресурсов, представляемых описателями, все объекты ядра используют их как унифицированный механизм управления доступом.

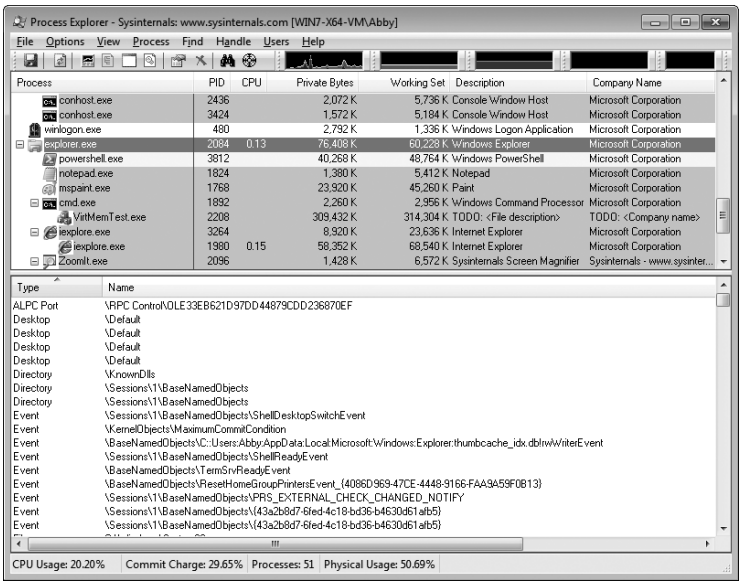


Рис. 3-23. Нижняя панель Procexp в режиме просмотра описателей

При попытке создать или открыть объект процесс запрашивает доступ для операций, которые он намеревается выполнять, например, доступ для чтения или записи. Если создать или открыть объект удалось, процесс запрашивает описатель к объекту, включающий предоставленные права доступа. Этот описатель может использоваться для дальнейших операций над объектом в рамках предоставленных прав доступа. Даже если пользователь имеет полный доступ к объекту, но запрошен был только доступ для чтения, соответствующий описатель пригоден только для операций чтения.

Хотя программы работают с описателем как с «черным ящиком», он является обычным целым числом. Оно является байтовым смещением в таблице описателей процесса, которая создается в памяти ядра. В таблицу описателей заносится тип объекта, предоставленный доступ и указатель на структуру данных (собственно, на объект).

Примечание Тем, кто программирует для Windows, знакомы типы данных из категории описателей, применяемых для управления объектами Диспетчера окон: HWND — для окон, HBRUSH — для кистей, HDC — для контекстов устройств и т. д. Эти объекты работают совсем иначе, чем рассматриваемые здесь описатели, никак с ними не связаны и не вносятся в таблицу описателей процесса.

Учтите, загрузка DLL или отображение другого типа файла в адресное пространство процесса обычно не приводит к добавлению описателя в таблицу описателей процесса. Следовательно, такие файлы могут использоваться и удалить их в этом случае невозможно, даже если найти соответствующие описатели не удается. Именно поэтому функция Find в Procexp ищет как DLL, так и описатели.

Для просмотра описателей, принадлежащих процессу, работающему в контексте безопасности, отличном от такового Procexp, необходимо запустить Procexp с правами администратора.

По умолчанию в режиме просмотра описателей отображаются типы и имена всех именованных объектов, открытых процессом, выбранным на верхней панели. Можно выбрать отображение дополнительной информации о каждом описателе, а также информации о безымянных объектах.

Настройка режима просмотра

Чтобы изменить набор столбцы, отображаемых в режиме просмотра описателей, нажмите Ctrl + H, чтобы войти в режим просмотра описателей, щелкните правой кнопкой заголовок столбца на нижней панели и выберите **Select Columns** — откроется окно **Select Columns** с вкладкой **Handle** (рис. 3-24).

Следующие атрибуты не изменяются, пока описатель открыт:

- **Type** Тип защищенного объекта, к которому описатель предоставляет доступ, в частности, рабочий стол, каталог, файл, раздел и поток.
- **Name** Имя объекта. Для большинства типов объектов в имя входит пространство имен объекта, например *\Device\Afd*. Для объектов файловой системы и реестра, корневых разделов вместо внутренних имен отображаются буквы диска и понятные имена, такие как \Device\HarddiskVolumel

(C:) и \REGISTRY\MACHINE\Software\Classes (HKCR). В описатель процесса входит имя процесса и PID; в имени описателя потока к этим сведениям прибавляется идентификатор потока (TID). В имя описателя маркера входят идентификаторы участника системы безопасности и сеанса входа. Безымянные описатели по умолчанию не отображаются.

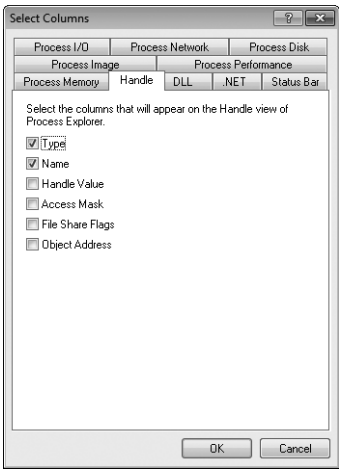


Рис. 3-24. Вкладка Handle в окне Select Columns

- **Handle Value (Значение описателя)** Значение описателя в шестнадцатеричной системе счисления, которое процесс передает API-функции для обращения к объекту. Это значение является байтовым смещением в таблице описателей процесса.
- **Access Mask (Маска доступа)** Битовая маска (в шестнадцатеричном формате), определяющая, какие разрешения процесс получает через описатель. Каждый установленный бит означает предоставленное разрешение для объекта. Так, разрешение «чтение» для раздела реестра выглядит как 0x00020019, а для файла — 0x00120089. Разрешение «полный доступ» для раздела реестра — 0x000F003F, а для файла — OxOOlFOlFF (подробнее см. в материалах о правах и масках доступа в библиотеке MSDN).
- **File Share Flags (Флаги общего доступа к файлу)** Режим общего доступа к объекту-файлу, установленный при открытии описателя. Флаги могут содержать буквы R, W или D, которые показывают, что другим программам (включая другие потоки того же процесса) разрешено открывать файл, соответственно, для чтения, записи или удаления. Если ни один флаг не установлен, с таким описателем объект файловой системы открывается для монопольного доступа.
- **Object Address (Адрес объекта)** Адрес структуры данных, представляющий объект, в памяти ядра. Эта информация может использоваться отладчиком ядра для вывода дополнительной информации об объекте.

Если в меню **View** выбрать пункт **Show Unnamed Handles And Mappings,** то в режиме просмотра описателей будут отображаться объекты, не имеющие связанного имени (учтите, что некоторые типы объектов всегда безымянны, другие же являются безымянными лишь время от времени). Безымянные объекты обычно создаются процессом для собственного использования. Они могут также наследоваться и использоваться дочерними процессами, если те могут определить, какой из унаследованных описателей следует использовать. Описатели могут также копироваться из одного процесса в другой, при условии что процесс, копирующий описатель, обладает достаточным уровнем доступа к процессу, из которого копируется описатель.

Примечание Procexp потребляет значительно больше ресурсов ЦП, если активен параметр **Show Unnamed Handles And Mappings**.

В режиме просмотра описателей главное меню содержит пункт **Handle** с командами **Properties** и **Close Handle**, последняя закрывает описатель. Обычно эта операция рискованна: процесс, которому принадлежит описатель, не «знает», что описатель закрыт, поэтому использование данной функции может привести к повреждению данных или краху приложения. Закрытие описателя в процессе **System** или критическом процессе пользовательского режима, например Csrss, может привести к краху системы.

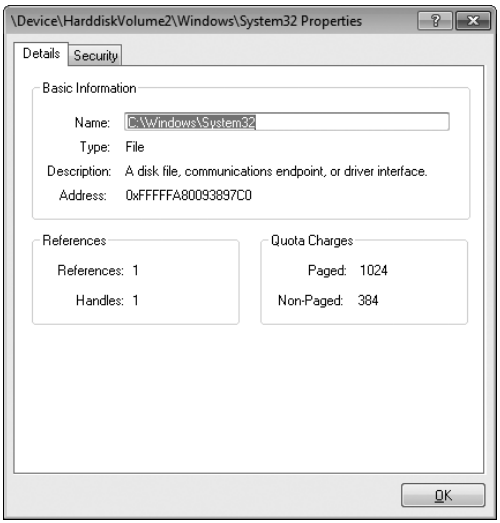


Рис. 3-25. Окно свойств описателя

Если дважды щелкнуть описатель или выбрать **Properties** в меню **Handle**, откроется окно **Properties** для данного описателя. На вкладке **Details** (рис. 3-25) отображаются внутреннее имя объекта, а в поле **Name** — его понятное имя. Так, имена \Device\HarddiskVolume2\Windows\ System32 и C:\ Windows\System32 равноценны. В окне также содержится более подробное описание объекта с коротким именем. В секции **References (Ссылки)** ото-

бражаются оставшиеся открытыми описатели и ссылки на объект. Поскольку каждый описатель содержит ссылку на объект, число ссылок никогда не бывает меньше числа описателей.

Разница между этими двумя значениями дает число прямых (а не опосредованных описателем) ссылок на объект из кода режима ядра. Объект может быть закрыт, только когда число ссылок на него уменьшится до нуля, то есть каждая открытая ссылка должна быть закрыта. Секция **Quota (Квота)** показывает, какую часть своей квоты выгружаемого и невыгружаемого пулов израсходовал процесс при создании объекта.

Вкладка **Security (Безопасность)** в окне **Handle Properties** представляет собой стандартное окно редактора системы безопасности, отображающее дескриптор защиты объекта, на который ссылается описатель. В некоторых случаях, особенно при работе с безымянными объектами, окно содержит предупреждение о риске для безопасности при отсутствии у объекта присвоенных разрешений. Для безымянных объектов это, как правило, не важно, поскольку в отсутствие имени другой процесс может получить доступ к объекту только через существующий описатель.

Сведения о процессе

Список процессов в главном окне Procexp может отображать огромное количество информации о всех процессах в системе, но можно получить еще больше подробностей о том или ином процессе. Для этого дважды щелкните его в главном окне Procexp — откроется окно свойств процесса. В нем Procexp отображает сведения на вкладках: **Image**, **Performance**, **Performance Graph**, **Threads**, **TCP/IP**, **Security**, **Environment** и **Strings**. При работе с правами администратора добавляется вкладка **Disk And Network (Диск и сеть)**. Дополнительные вкладки добавляются для процессов–служб, связанных с заданием или использующих .NET Framework.

Окно **Properties** не является модальным, поэтому можно, не закрывая его, переключаться на главное окно. Можно открывать сразу несколько окон **Properties**, изменять их размер и разворачивать на весь экран.

Для получения основной информации, отображаемой в окне свойств процесса, нужен полный доступ к этому процессу либо возможность определить полный путь к его образу. Без прав администратора Procexp сможет отображать подробную информацию только о процессах, работающих под той же учетной записью пользователя. В отличие от вкладки **Disk And Network**, которая всегда требует прав администратора, здесь имеется несколько исключений.

Вкладка Image

Вкладка **Image** (рис. 3-26) отображает информацию о процессе, которая почти не изменяется в течение жизненного цикла, в том числе сведения из внедренных ресурсов: значок, версия образа, полный путь к образу, команд-

ная строка, использованная для запуска процесса, учетная запись, под которой работает процесс, время запуска, включены ли DEP и ASLR (только в Windows Vista и выше), а в x64 версиях Windows — еще и тип кода: 32- или 64-разрядный. Два поля, которые могут изменяться — текущий каталог и родительский процесс. Если родительский процесс еще работает во время запуска Procexp, в соответствующем поле будут его имя образа и PID, если он завершился, в поле будет строка <Non-existent Process> (Несуществующий процесс) и PID.

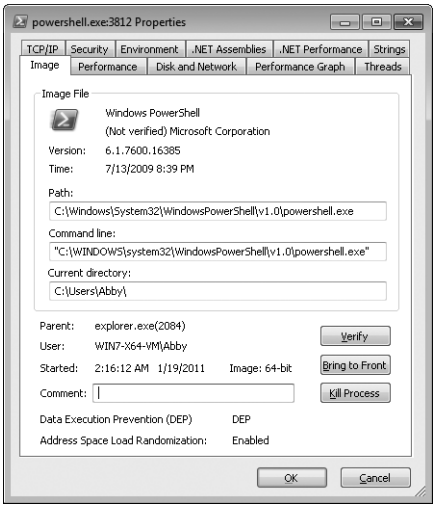


Рис. 3-26. Вкладка Image в окне свойств процесса

Второе поле на вкладке **Image** — аналог поля **Verified Signer**, оно отображает имя компании, взятое из образа, или имя подписавшего файл владельца проверенного сертификата. Если проверка подписи не выполнялась, можно щелкнуть кнопку **Verify**, чтобы выполнить ее.

Если у процесса есть видимое окно на текущем рабочем столе, то щелчок кнопки **Bring To Front (Активировать)** открывает это окно поверх остальных. Если у процесса несколько видимых окон, то кнопка **Bring To Front** активирует окно, лежащее поверх остальных.

Щелчок кнопки **Kill Process** приводит к принудительному завершению процесса. По умолчанию Procexp запросит подтверждение, но можно отключить запрос, сняв флажок **Confirm Kill** в меню **Options**.

Внимание! Принудительная остановка не дает процессу возможности корректно завершить работу и может привести к потере данных или нестабильности системы. Кроме того, Procexp не предупреждает о попытке остановки критически важных для системы процессов, например Csrss.exe. Остановка критического процесса приводит к немедленному краху Windows с «синим экраном».

Добавить комментарий к процессу можно в поле **Comment**. Комментарии будут видны в списке процессов, если отображается столбец **Comment**, а если он скрыт, то во всплывающей подсказке процесса. Комментарий отно-

сится ко всем процессам, имеющим одинаковый путь, и сохраняются при последующих запусках Procexp. Имейте в виду, что для идентификации пути к образу для процессов, работающих под другими записями, нужны права администратора. Если путь к образу узнать невозможно, используется имя процесса. Это означает, например, что комментарий для процесса svchost.exe при работе Procexp с правами администратора будет связан и с «C:\Windows\System32\svchost.exe», а в отсутствие прав администратора — только с «svchost.exe», но не с процессом, для которого указан полный путь. Procexp сохраняет комментарии в том же разделе реестра, что и другие настройки конфигурации (HKCU\Software\Sysinternals\Process Explorer).

Вкладка Performance

Вкладка **Performance** (рис. 3-27), отображает счетчики загруженности ЦП, виртуальной памяти, физической памяти (рабочего набора), ввода-вывода, описателей объектов ядра и Диспетчера окон. Все данные на этой вкладке обновляются при обновлении окна Procexp.

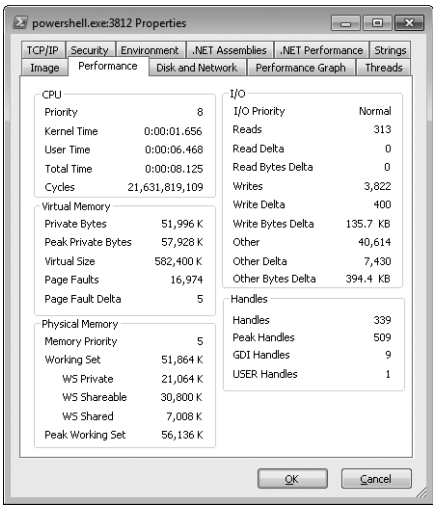


Рис. 3-27. Вкладка Performance в окне свойств процесса

Вкладка **Performance** удобна тем, что позволяет увидеть множество показателей процесса в одном месте. Большинство полей вкладки **Performance** можно вывести и в списке процессов, как было сказано выше. Поля вкладки **Performance** описаны выше в разделах о вкладках **Process Performance**, **Process Memory** и **Process I/O** окна **Select Columns**. Дополнительные секции, отображаемые только на вкладке **Performance**, показывают долю времени в режимах ядра и пользователя в загруженности ЦП, а также максимальное количество описателей.

Вкладка Performance Graph

Вкладка **Performance Graph** отображает графики параметров производительности, подобные графикам Диспетчера задач, но для отдельного процесса (рис. 3-28). Верхний график показывает изменения загруженности ЦП, где красным обозначена загруженность в режиме ядра, а зеленым — в режиме пользователя. Если навести указатель мыши на график, появится подсказка с указанием процента времени, затраченного этим процессом, от общего затраченного времени ЦП от общего времени ЦП на момент, соответствующий точке под указателем, а также время. Имейте в виду, что этот график отображает сводные данные для всех процессоров. Если процесс загрузил один из ЦП двухпроцессорной системы на 100%, оставив второй свободным, на графике будет значения загруженности 50%.

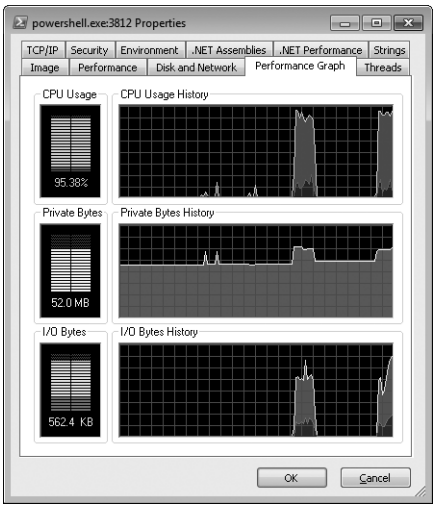


Рис. 3-28. Вкладка Performance Graph в окне свойств процесса

Второй график показывает изменение количества переданной процессу закрытой памяти и масштабируется по максимальному значению: если максимум увеличивается, график масштабируется по новому значению. Если навести указатель мыши на график, отображается счетчик закрытой памяти и время для точки под указателем. Постоянный рост значений на этом графике может свидетельствовать об утечке памяти.

Третий график отображает интенсивность файлового ввода-вывода файла и ввода-вывода на устройствах, осуществляемого процессом. Голубая линия показывает суммарный поток данных ввода-вывода, а розовая — поток данных при записи. График ввода-вывода масштабируется по максимальному значению, зарегистрированному с начала мониторинга. Если навести указатель мыши на график, отображается количество прочитанных и записанных байтов; байтов, переданных в других операциях, и время для точки под указателем.

Как сказано выше, допускается изменять размер диалогового окна и разворачивать его на весь экран. Чем шире окно, тем дольше интервал, отображаемый на графиках.

Вкладка Threads

Вкладка **Threads** в окне свойств процесса отображает подробную информацию о его потоках, включая текущие стеки вызовов, а также позволяет останавливать или приостанавливать отдельные потоки процесса. Подробнее об этом будет рассказано ниже.

Вкладка TCP/IP

Все активные конечные точки соединений по протоколам TCP, TCPV6, UDP, или UDPV6, принадлежащие процессу, показаны в списке на вкладке **TCP/IP** (рис. 3-29). Для каждого соединения отображаются протокол, состояние, а также локальный и удаленный адреса и номера портов. Для процессов служб Windows Vista и выше на вкладку добавляется столбец **Service** для служб — адресов подключений. Установите флажок **Resolve Addresses (Определять адреса)**, чтобы определить адреса конечных точек соединений по DNS-именам. Если снять этот флажок, будут отображаться адреса IPv4 или IPv6.

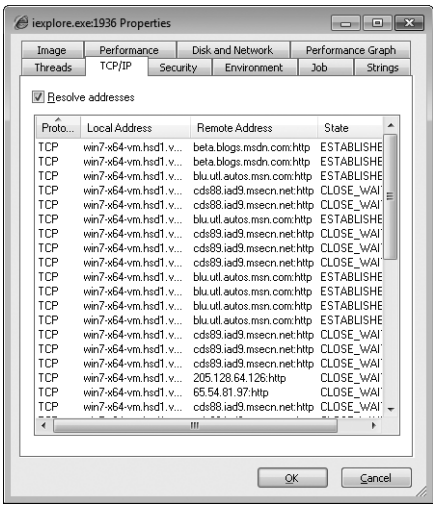


Рис. 3-29. Вкладка TCP/IP в окне свойств процесса

В Windows XP на этой вкладке находится кнопка **Stack (Стек)**, открывающая окно со стеком потока, открывшего выделенную конечную точку (рис. 3-30). Это удобно для определения конечных точек соединений, установленных процессами **System** и **Svchost**, поскольку стек содержит имя драйвера или службы, открывающих соединение.

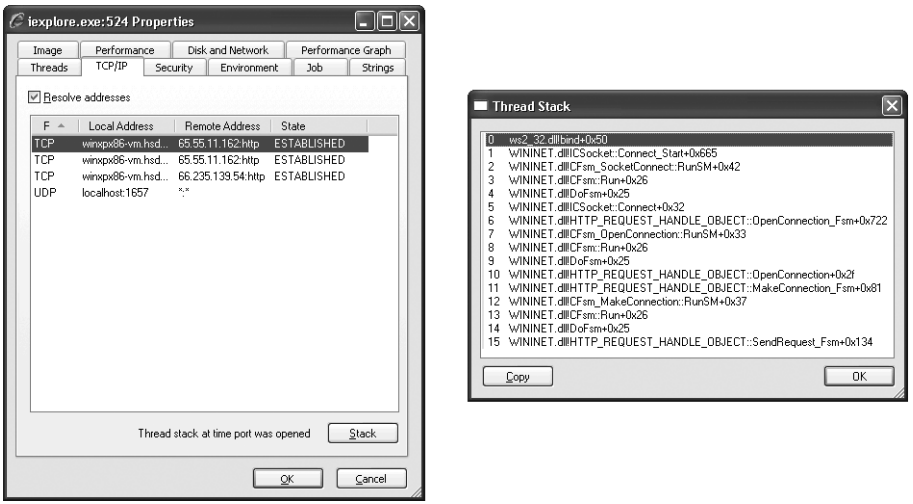


Рис. 3-30. Вкладка TCP/IP в Windows XP и в окне Thread Stack

Вкладка Security

Маркер процесса определяет его контекст защиты: базовое имя пользователя, под записью которого работает процесс; группы, в которые входит эта запись, и ее привилегии в системе. Вкладка **Security** (рис. 3-31) отображает все эти сведения, а также показывает, включена ли для процесса виртуализация UAC для файлов и реестра, выводит также ID сеанса служб терминалов, в котором работает процесс. При выборе группы в списке **Group** под списком отображается идентификатор защиты (SID) выбранной группы.

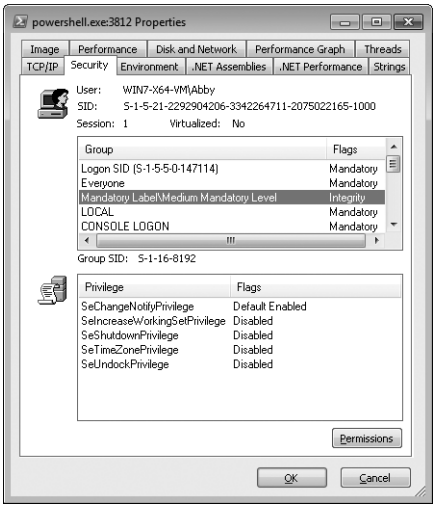


Рис. 3-31. Вкладка Security в окне свойств процесса

Обычно (особенно в случае «настольных» приложений) проверка прав доступа выполняются по маркеру процесса или (иногда) маркеру потока,

производному от маркера процесса. Таким образом, у маркера потока не может быть больше прав, чем у маркера процесса. Информация вкладки **Security** помогает найти причину успеха или неудачи различных операций.

Службы и серверные приложения могут олицетворять контекст защиты другого пользователя при выполнении действий от имени этого пользователя. Олицетворение осуществляется путем связывания копии маркера олицетворяемой записи пользователя с одним из потоков процесса. Во время олицетворения проверка прав доступа выполняются по маркеру потока, маркер процесса в таких случаях применяться не может. Вышеперечисленные окна не отображают маркеры потоков.

Не буду углубляться в устройство маркера, но дам несколько полезных советов, чтобы развеять некоторые распространенные заблуждения:

- группа, для которой установлен флаг **Deny (Запрет)**, равноценна отсутствию данной группы в маркере. Если активирован UAC, то группы с высоким уровнем прав, например **Администраторы** помечены флагом **Deny-Only (Только запрет)**, за исключением процессов с повышенными привилегиями. Флаг **Deny** говорит о том, что при наличии в списке управления доступом (ACL) объекта разрешающего доступ элемента (ACE) для группы **Администраторы** этот элемент игнорируется, а при наличии ACE, запрещающего доступ **Администраторам** (а не всем вообще), доступ будет запрещен;
- привилегия, отмеченная флагом **Disabled (Отключена)**, вовсе не считается отсутствующей. Напротив, если привилегия фигурирует в маркере, программа может активировать ее и использовать. Если же привилегия отсутствует вообще, процесс никак не сможет получить ее. Важно отметить, что в Windows Vista и выше ряд привилегий приравнивается к администраторским. Windows никогда не вносит эти привилегиям в маркеры обычных пользователей;
- если компьютер, подключенный к домену, не может связаться с контроллером домена и не кешировал результаты прежних запросов разрешения SID в имя, он не может установить имена групп по их SID. В этом случае Procexp просто отображает SID;
- SID группы Logon основан на случайном числе, сгенерированном в момент входа пользователя в систему. Одна из его функций — предоставление доступа к специальным ресурсам сеанса служб терминалов. SID группы Logon всегда начинаются с S-1-5-5-.

Кнопка **Permissions (Разрешения)** отображает дескриптор защиты самого объекта процесса, то есть показывает, кто какие действия может выполнять с процессом.

Вкладка Environment

Вкладка **Environment (Окружение)** отображает переменные окружения процесса и их значения. Процессы обычно наследуют свои переменные окру-

жения от родительских процессов, и зачастую блоки переменных окружения всех процессов оказываются, в сущности, эквивалентными. Исключения возникают, когда:

- родительский процесс может определять другой набор переменных окружения для дочернего процесса;
- процесс добавляет, удаляет или изменяет свои переменные окружения (это может каждый процесс);
- не все процессы (особенно консольные программы) принимают широковещательные сообщения, оповещающее работающие процессы об изменении конфигурации переменных окружений системы, поэтому не все процессы вносят эти изменения в свои блоки переменных окружения.

Вкладка Strings

Вкладка **Strings (Строки)** в окне свойств процесса (рис. 3-32) показывает все последовательности из трех и более пригодных для печати знаков, найденных в образе процесса; подробнее про отображение строк рассказывается выше в этой главе.

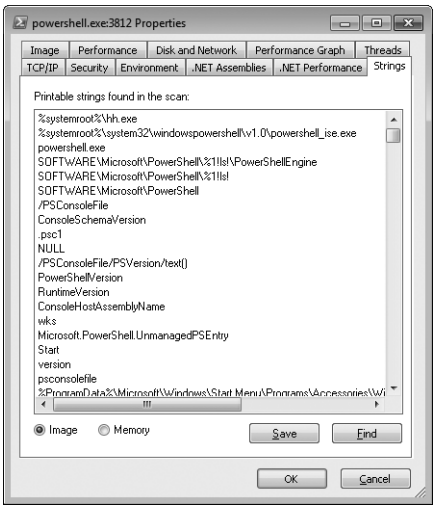


Рис. 3-32. Вкладка Strings в окне свойств процесса

Вкладка Services

Службы Windows работают в процессах (как правило, неинтерактивных), которые можно настроить так, чтобы они запускались независимо от входа в систему пользователя, а также для управления через стандартный интерфейс SCM (Диспетчера управления службами). Возможен запуск нескольких служб в одном процессе, типичным примером является Svchost.exe, разработанный специально для хостинга нескольких служб, реализованных в отдельных DLL.

Если выбранный процесс содержит одну или несколько служб, в окно свойств процесса добавляется вкладка **Services** (рис. 3-33). Список на этой вкладке содержит внутренние и отображаемые имена каждой службы, а также служб, содержащихся в процессах Svchost.exe, и пути к их DLL. При выборе службы в списке ниже открывается ее описание.

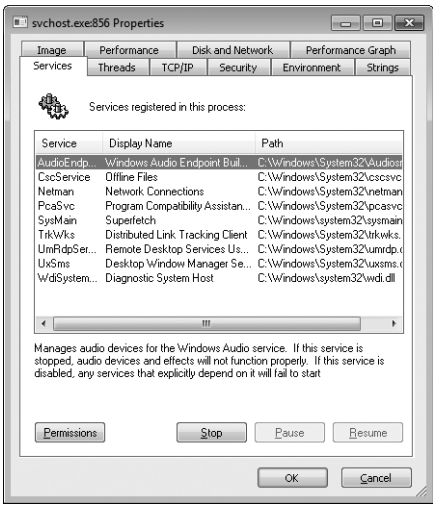


Рис. 3-33. Вкладка Services в окне свойств процесса

Для отдельных служб можно разрешать либо запрещать остановку, приостановку и возобновление работы. Procexp отображает кнопки **Stop**, **Pause** и **Resume**, если данные операции разрешены для выбранной службы.

Кнопка **Permissions** отображает окно редактора защиты и позволяет просматривать и изменять разрешения для службы. Отдельные разрешения для служб включают **Start (Пуск)**, **Stop (Остановка)**, **Pause/Resume (Пауза/Возобновление)**, **Query Status (Запрос состояния)**, **Query Config (Запроса конфигурирования)**, **Change Config (Изменение конфигурации)**, **Interrogate (Опрос)**, **Enumerate Dependents (Перечисление зависимостей)**, **User-Defined Control (Управление пользователем)**, а также стандартные **Read (Чтение)**, **Change Permissions (Изменение разрешение)** и **Change Owner (Смена владельца)**.

Внимание! Предоставив разрешение Write (даже без прав администратора) либо отдельных разрешений Change Config, Change Permissions или Change Owner какой-либо службе позволит постороннему без труда получить полный администраторский доступ к компьютеру.

Вкладки .NET

Если выбранный процесс использует .NET Framework, Procexp добавляет в окно свойств процесса вкладки **.NET**. Вкладка **.NET Performance (Производительность .NET)** (рис. 3-34) отображает домены приложений

в процессе и данные девяти наборов счетчиков производительности .NET. Выберите объект производительности из раскрывающегося списка (например, **.NET CLR Data**, **Exceptions**, **Interop**, **Memory** или **Security**) — в списке ниже откроется набор текущих счетчиков объекта.

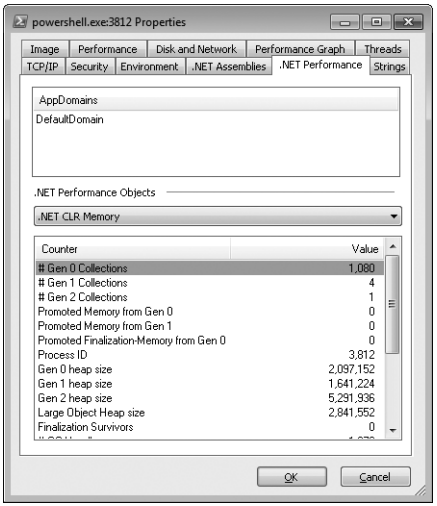


Рис. 3-34. Вкладка .NET Performance в окна свойств процесса

Когда Procexp работает с правами администратора в Windows Vista и выше, на вкладке **.NET Assemblies (Сборки .NET)** (рис. 3-35) отображаются все домены приложений в процессе, включая дерево имен загруженных в них сборок. Справа от имени Procexp показывает флаги и полный путь к исполняемому образу сборки. Для получения этой информации Procexp использует недокументированные события .NET ETW.

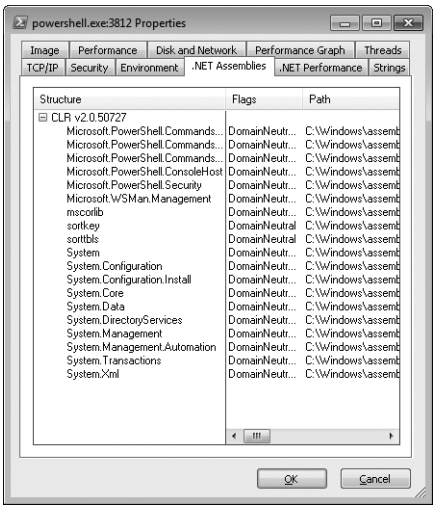


Рис. 3-35. Вкладка .NET Assemblies в окне свойств процесса

Вкладка Job

Если выбранный процесс связан с заданием, Procexp добавляет в окно свойств процесса вкладку **Job**. Вкладка показывает имя задания, если оно имеется, процессы в задании, а также все ограничения, наложенные на задание. На рис. 3-36 хост-процесс WMI связан с заданием, включающим еще один хост-процесс WMI. Задание разрешает передавать каждому процессу не более 512 Мб памяти, но не более 1 Гб в сумме; в задании может быть до 32 процессов, работающих одновременно.

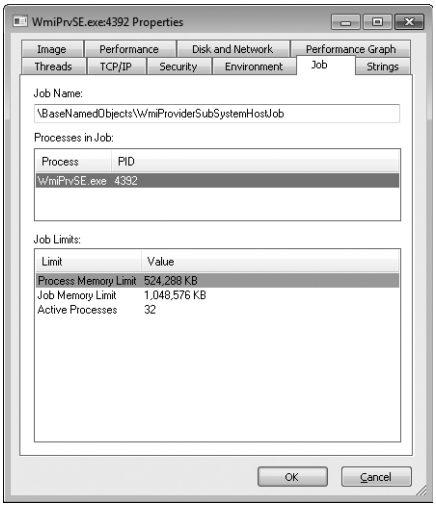


Рис. 3-36. Вкладка Job в окне свойств процесса

Сведения о потоках

Как сказано выше, процесс в действительности не исполняют код, а является контейнером для ресурсов, к которым относятся виртуальное адресное пространство, спроецированные образы, содержащие код для исполнения, а также потоки исполнения. Поток — это объект, реально исполняющий код. Его ресурсы включают стек вызовов и указатель на следующую исполняемую команду (подробнее см. в главе 2).

Вкладка **Threads** в окне свойств процесса (рис. 3-37) отображает подробную информацию о каждом потоке в текущем процессе. В списке наверху окна отображается следующая информация:

- **TID** Назначенный системой уникальный идентификатор потока. Идентификатор потока можно использовать повторно после завершения «носившего» его потока, но потоки с одинаковыми TID не могут сосуществовать в системе.
- **CPU** Процент общего времени ЦП, затраченного потоком с момента предыдущего обновления. Имейте в виду, что поток не может использовать более 100% времени одного логического ЦП, поэтому на двухъядер-

ном ПК это число не может превышать 50%, на четырехъядерном — 25% и т. д.

- **Cycles Delta or C Switch Delta (Прирост времени или числа переключений контекста)** Если Procexp работает в Windows Vista и выше в контексте, предоставляющем полный доступ к процессу, в этом поле будет прирост затраченного времени ЦП. Это количество времени ЦП, использованного потоком с момента предыдущего обновления. Прирост числа переключений контекста показывает, сколько раз поток был подключен к ЦП с момента предыдущего обновления.
- **Service** Отображается в Windows Vista и выше для процессов, содержащих одну или несколько служб; показывает, какая служба с каким потоком связана. Windows помечает потоки процессов-служб, чтобы связать конечные точки потоков и TCP/IP-соединений с установившими их службами.
- **Start Address (Стартовый адрес)** Символическое имя указанного программой адреса виртуальной памяти процесса, с которого началось исполнение потока (в формате Имя_модуля!Имя_функции, подробнее см. в главе 2). Если в Procexp настроен доступ к веб-серверу символов, то отображение этой вкладки может задерживаться в ожидании загрузки требуемых символов, в этом случае над списком отображается индикатор хода загрузки.

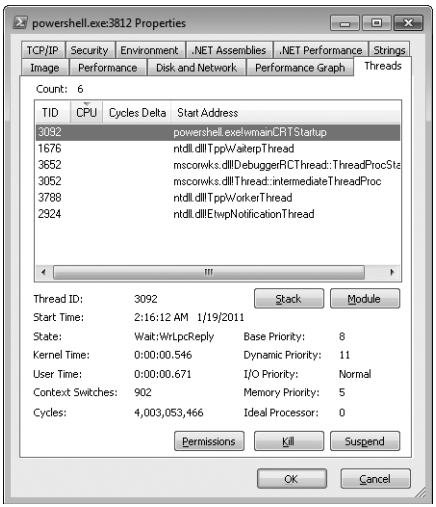


Рис. 3-37. Вкладка Threads в окне свойств процесса

По умолчанию список сортируется по времени ЦП в порядке убывания. Для изменения порядка сортировки щелкните заголовок любого столбца, можно изменять размеры столбцов, но не их порядок.

При выборе строки в списке на нижней панели вкладки **Threads** выводится более подробная информация о потоке: время запуска, загруженность ЦП в режимах ядра и пользователя, число переключений контекста и использо-

ванное время ЦП, базовый и динамический приоритеты. В Windows Vista и выше также отображаются приоритеты ввода-вывода и памяти, а также идеальный процессор. Щелчок кнопки **Permissions** открывает дескриптор защиты потока, который показывает, кому какие действия разрешено выполнять с потоком. Интерфейс вкладки позволяет изменять разрешения потока, но делать это не рекомендуется, поскольку такие действия приводят к непредсказуемым последствиям.

Для процесса **System Idle** вместо потоков отображаются процессоры. Вместо ID потока показан номер процессора, а время ЦП означает процент времени бездействия данного ЦП с момента последнего обновления. При выборе одного из процессоров в списке вместо времени в режиме ядра ниже списка выводится общее время бездействия данного ЦП.

Щелчок кнопки **Module (Модуль)** открывает стандартное окно свойств файла для EXE- или DLL-файла, фигурирующего в выбранной строке.

Кнопка **Stack** показывает стек вызовов выбранного потока (рис. 3-38). Стартовый адрес расположен внизу стека, а текущий вызов потока — вверху. Кнопка **Copy** в окне **Stack** копирует выбранное символьное из стека в буфер обмена. Можно выбрать несколько строк стандартными способами, например, удерживая клавишу **Shift**, нажимать клавиши-стрелки (подробнее см. в главе 2).

Рис. 3-38. Стек вызовов потока

Наконец, кнопки **Kill** и **Suspend** позволяют уничтожить или приостановить выбранный поток. Пока вы не разобрались, что именно делает процесс (если, конечно, вы не автор этой программы), не стоит уничтожать или приостанавливать потоки в процессе.

Проверка подписей образов

В исполняемый файл могут внедряться ресурсы со сведениями о версии, полями **Company Name, Description, Copyright** и прочей информацией об издателе файла, но ее наличие не гарантирует ее подлинности: любой может написать программу и ввести в поле **Company Name** строку «Microsoft». Цифровая подпись файла позволяет убедиться в том, что файл действительно был создан тем, кто на нем «расписался», и не был изменен позднее.

Procexp может проверять подлинность цифровых подписей исполняемых файлов и DLL в анализируемых процессах. По умолчанию проверка выполняется только по запросу. На вкладке **Image** в окнах свойств процесса и DLL имеется кнопка **Verify**, проверяющая подлинность и целостность образов и DLL. Можно также автоматически выполнять проверку подписей для всех файлов. Для этого нужно выбрать пункт **Verify Image Signatures** в меню **Options**. Помимо окон свойств, состояние проверки подписи можно увидеть и в главном окне, включив отображение столбца **Verified Signer** в основном режиме и в режиме просмотра DLL.

Если подпись выбранного файла проверена, в поле состояния проверки появляется строка **Verified** и имя владельца сертификата, которым файл подписан. Если проверка не выполнялась (или выбранный файл не является исполняемым), это поле остается пустым или содержит строку **Not verified** и название компании, взятое из файла. Если файл не подписан или проверка не прошла, в поле появляется строка **Unable to verify** и название компании.

Имейте в виду, что имя в сертификате подписи и название компании, внедренное в файл, могут не совпадать. Например, в большинство исполняемых файлов, поставляемых с Windows, внедрено название издателя «Microsoft Corporation», но подписаны они сертификатом «Microsoft Windows».

К типичным причинам, по которым не удается проверить цифровую подпись, относятся следующие:

- файл не подписан;
- файл изменен после подписания;
- сертификат подписи не отслеживается до корневого центра сертификации, доверяемого на данном компьютере (это вполне вероятно, если обновление сертификата корневого ЦС отключено групповой политикой);
- сертификат подписи аннулирован.

Окно System Information

Диалоговое окно **System Information** (рис. 3-39) аналогично вкладке **Быстродействие** Диспетчера задач**,** однако содержит гораздо больше информации. Чтобы открыть его, нажмите Ctrk+l или щелкните любой миниграфик на панели инструментов главного окна.

Вкладка **Summary (Сводка)** содержит несколько графиков системных счетчиков, подробнее описанных при обсуждении вкладок **CPU**, **Memory**, **I/O** (рис. 3-40, 3-41 и 3-42, соответственно). Левое поле в каждой паре показывает текущее значение счетчика, а широкий график справа — его изменения за последнее время. Чем шире диалоговое окно, тем больший период охватывает график. Если навести указатель мыши на графики, откроется подсказка с указанием времени и значения счетчика для точки графика под указателем. Подсказка к графикам **CPU Usage** и **I/O** также показывает процесс, сильнее всего загружавший ЦП в тот момент. Щелчок любого из графиков «замораживает» подсказку: графики продолжают обновляться, но

содержимое подсказки не изменится, пока вы не уберете указатель мыши с графика.

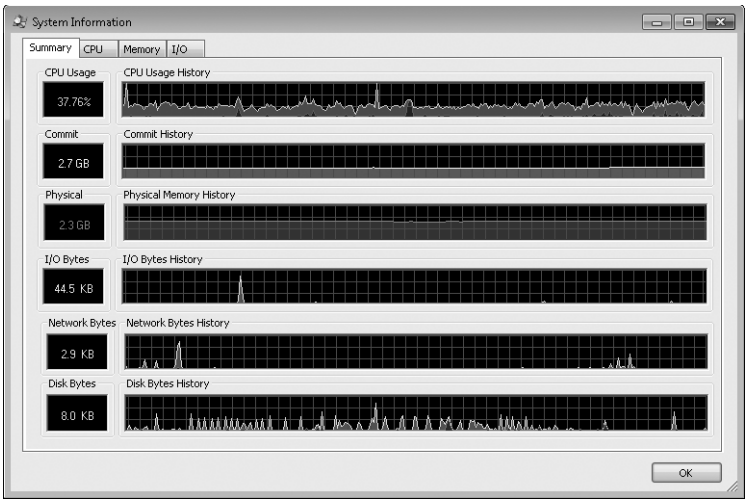


Рис. 3-39. Вкладка Summary в окне System Information

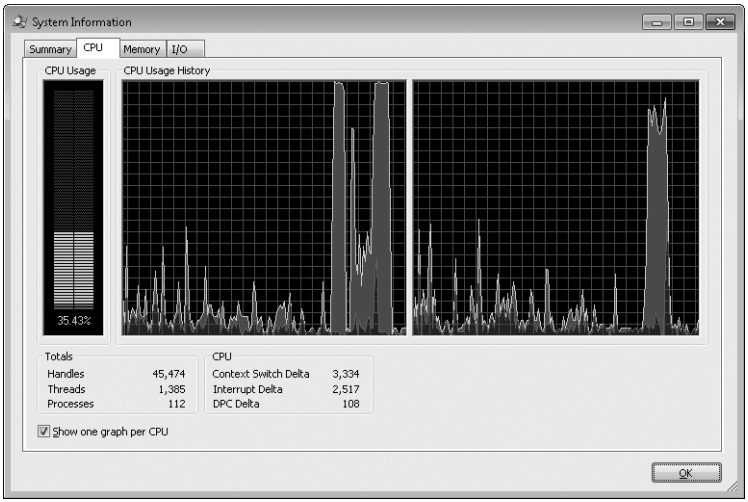


Рис. 3-40. Вкладка CPU в окне System Information

На графиках **CPU Usage** красная область показывает процент затраченного времени ЦП в режиме ядра, а область под зеленой линией — общую загруженность ЦП в процентах. Если компьютер имеет несколько логических ЦП, то при установке флажка **Show One Graph Per CPU (Показать отдельный график для каждого ЦП)** в левом нижнем углу вкладки **CPU** график **CPU Usage History (История загруженности ЦП)** на этой вкладке делится на отдельные графики для каждого ЦП. Графики ЦП всегда отображаются в шкале 100%. Имейте в виду: если выводится несколько графиков

ЦП, всплывающая подсказка на **CPU Usage** показывает процесс, сильнее всего загружающий ЦП во всей системе (Procexp не отслеживает, какой из процессов использует наибольшее количество времени ЦП). Также учтите, что при отображении графиков для отдельных ЦП Procexp приходится использовать измерения по таймеру, поскольку такты отдельных ЦП не отслеживаются Windows. В итоге график отдельных ЦП может показывать иные результаты, чем сводный график и главное окно Procexp.

В нижней части вкладки **CPU** отображается общее число открытых описателей объектов, потоков и процессов в системе, а также число переключений контекста ЦП, прерываний и DPC с момента предыдущего обновления.

Вкладка **Memory** показывает графики переданной **(Commit)** и физической памяти **(Physical Memory)**. На первом графике область под желтой линией отражает объем переданной памяти — общее объем закрытой памяти (в байтах), выделенной процессам, плюс выгружаемый пул. График масштабируется по лимиту выделения — максимального объема закрытой памяти, который может быть передан без увеличения размера файла подкачки. График отображает серию «снимков» системы, сделанных при обновлении, но не показывает, что происходит в интервалах между обновлениями. Например, если в момент обновления Procexp передано 1 Гб, затем создается процесс, который выделяет 1,5 Гб памяти и высвобождает ее до следующего обновлениея данных Procexp, объем переданной памяти на графике останется прежним — 1 Гб, пика зарегистрировано не будет. Графики утилизации физической памяти показывают объем физической оперативной памяти, используемый системой. Этот объем сопоставляется с объемом физической памяти, установленным на компьютере и доступным для Windows. Подобно графику переданной памяти, график физической памяти представляет собой последовательность «снимков» системы и ничего не сообщает об изменениях между обновлениями.

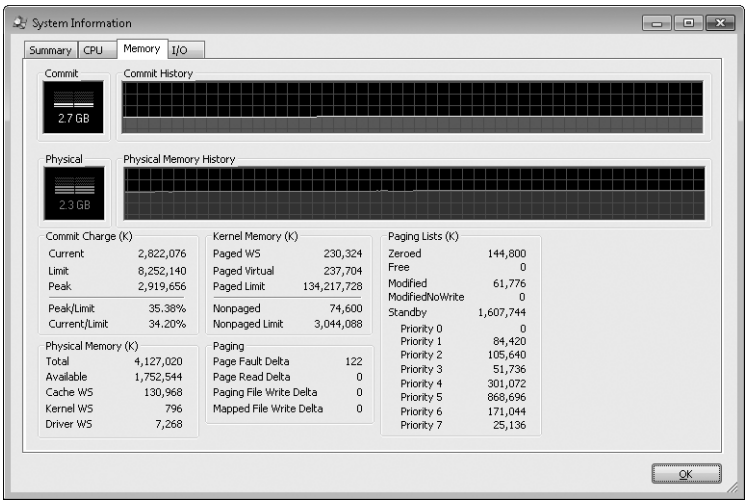


Рис. 3-41. Вкладка Memory в окне System Information

Нижняя часть вкладки **Memory** отображает ряд показателей, относящихся к памяти:

- **Commit Charge (K) (Выделенная память (ядра))** Текущий объем переданной памяти; предел, свыше которого нельзя выделить закрытую память без увеличения размера файла подкачки, а также пиковое значение переданной памяти с момента последней загрузки системы. Здесь же отображается проценты пикового значения от предельного и текущего значения от предельного.
- **Physical Memory (K) (Физическая память (ядра))** Общий объем физической памяти **(**в Кб), доступный для Windows, доступная свободная оперативная память, а также размеры рабочих наборов кеша, ядра и драйверов.
- **Kernel Memory (K) (Память ядра (ядро))** Выгружаемый рабочий набор представляет собой объем выгружаемого пула в килобайтах, присутствующий в оперативной памяти. **Paged Virtual (Выгружаемая виртуальная память)** — общий объем памяти, переданной для выгружаемого пула, включая байты, сброшенные в файл подкачки. **Paged Limit (Предел выгружаемого пула)** — максимальный объем памяти, который система позволяет выделить для выгружаемого пула. **Nonpaged (Невыгружаемый)** — объем памяти, который может быть выделен для невыгружаемого пула. Procexp требует прав администратора и символов для правильного отображения значений **Paged Limit** и **Nonpaged Limit**.
- **Paging (Постраничная подкачка)** Число страничных ошибок с момента предыдущего обновления данных, количество число операций чтения проецируемого или страничного файла при подкачке страниц, число операций записи в страничный файл или проецируемые файлы.
- **Paging Lists (K) (Списки подкачки (ядро))** **(**Только в Windows Vista и выше) Объем памяти (в килобайтах) в различных списках страниц, поддерживаемых Диспетчером памяти.

Вкладка **I/O** показывает счетчик **I/O Bytes** и, если Procexp работает с правами администратора, **Network Bytes** и **Disk Bytes**. Счетчик **I/O Bytes** показывает суммарную интенсивность ввода-вывода (файлового и на устройствах), **Network Bytes** отражает сетевой ввод-вывод, а **Disk Bytes** — трафик ввода-вывода на локальных дисках. Все они масштабируются по максимальному значению с начала мониторинга. Трафик операций записи обозначен красным, а общий трафик ввода-вывода — голубым. В отличие от графика выделения памяти, графики ввода-вывода учитывают события в период между обновлениями. Даже если приостановить обновление, при следующем обновлении на графике будет показана интенсивность ввода-вывода с учетом операций, выполненных во время паузы. При этом могут регистрироваться новые пики и даже меняться масштаб графика.

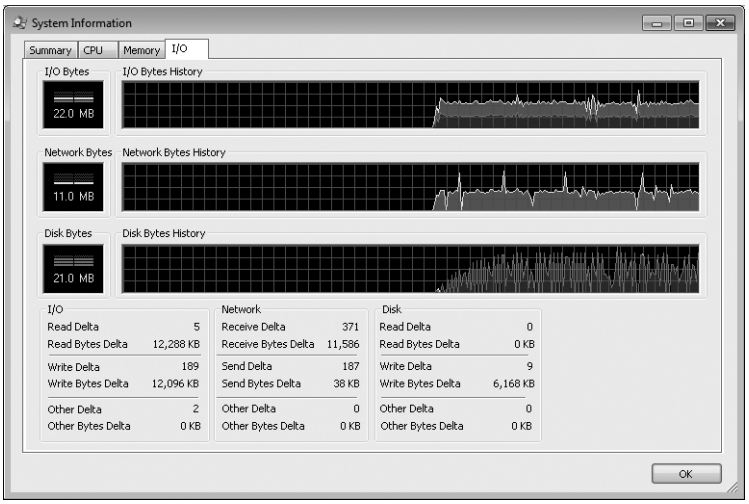


Рис. 3-42. Вкладка I/O в окне System Information

Нижняя часть вкладки **I/O** отображает общее число операций ввода-вывода, операций чтения, записи и других дисковых операций, а также получения, отправки и других сетевых операций с момента предыдущего обновления данных, а также количество байтов, прочитанных, записанных, переданных или принятых в данных операциях.

Параметры отображения

Помимо широких возможностей настройки отображаемого содержимого, описанных выше, Procexp предоставляет дополнительные параметры, о которых еще не говорилось в данной главе:

- **Hide When Minimized (Свернуть в трей)** Если активировать этот параметр в меню **Options**, Procexp показывает только значок в области уведомлений, когда программа свернута, и не показывает свернутое окно на панели задач. Кроме того, если щелкнуть стандартный значок-крестик в правом верхнем углу заголовка окна, то программа Procexp будет свернута, а не закрыта (как и Диспетчер задач).
- **Allow Only One Instance (Разрешать запуск только одного экземпляра)** Если запустить Procexp, выбрав данный параметр в меню **Options**, программа проверите, нет ли уже запущенного экземпляра Procexp на том же рабочем столе. Если таковой найден, новый экземпляр запущен не будет, а откроется старый.
- **Always On Top (Поверх всех окон)** Если выбрать данный параметр в меню **Options**, Procexp остается поверх всех окон (за исключением окон других программ с аналогичными настройками) на рабочем столе.
- **Font (Шрифт)** Данный элемент меню **Options** позволяет выбрать другой шрифт для главного окна и нижней панели, а также для многих элементов диалоговых окон **Procexp**.

- **Opacity (Непрозрачность)** Пункт **Opacity** в меню **View** позволяет устанавливать уровень прозрачности главного окна Procexp.
- **Scroll To New Processes (Прокрутить на новый процесс)** При выборе данного параметра в меню **View** Procexp прокручивает список процессов к новому процессу, только что появившемуся в списке.
- **Show Processes From All Users (Отображать процессы всех пользователей)** Этот параметр из меню **View** выбран по умолчанию. Он включает отображение всех процессов, работающих на данном компьютере. Если этот параметр неактивен, в списке процессов отображаются только процессы, работающие под той же учетной записью пользователя, что и Procexp. «Свои» процессы в этом случае цветом не подсвечиваются. У Диспетчера задач имеется похожая, но не идентичная функция. Одноименный параметр Диспетчера задач отличается тем, что позволяет отображать процессы, работающие в текущем сеансе либо во всех сеансах служб терминалов. Использование данного параметра Диспетчера задач также требует прав администратора.

Procexp как замена Диспетчера задач

Поскольку Procexp предоставляет гораздо больше полезной информации, нежели Диспетчер задач, можете пользоваться только этой программой, отправив Диспетчер задач «на покой». У Procexp имеется параметр, позволяющий сделать это. При выборе параметра **Replace Task Manager (Заменить Диспетчер задач)** в меню **Options** при вызове диспетчера задач в Windows будет запускаться Procexp вместо TaskMgr.exe независимо от способа вызова. Например, команда **Start Task Manager (Запустить Диспетчер задач)** из контекстного меню будет запускать Procexp, как и комбинация Ctrl+Shift+Esc.

Параметр **Replace Task Manager** имеет ряд особенностей:

- это глобальный параметр, касающийся всех пользователей компьютера. Если Procexp.exe находится в каталоге, недоступном другому пользователю, последний не сможет запустить ни Procexp ни Диспетчер задач.
- для установки данного параметра необходимы права администратора;
- этот параметр не изменяет и не удаляет Taskmgr.exe в папке System32. Вместо этого он использует **Image File Execution Options (Параметры исполнения файла-образа)**, чтобы сообщить Procexp.exe о запуске Taskmgr.exe.
- для восстановления возможности запуска Диспетчера задач выберите пункт **Restore Task Manager (Восстановить Диспетчер задач)** в меню **Options** (этот появляется в меню, когда Диспетчер задач отключен).

Создание процессов в Procexp

В Диспетчере задач новый процесс запускается с помощью меню **File | Run (Файл | Выполнить)**, в Procexp оно тоже есть**.** Кроме того, новый процесс с

повышенными или ограниченными правами можно создать с помощью других команд меню **File:**

- в Windows Vista и выше: если Procexp работает с обычными привилегиями, команда **Run As Administrator** требует повышения привилегий для запуска нового процесса;
- в Windows XP: **Запуск от имени** позволяет запустить новый процесс под любой другой учетной записью, для которой вы знаете учетные данные;
- **Run As Limited User (Запуск от имени пользователя с ограниченными правами)** запускает новый процесс с ограниченными правами. В Windows Vista и более поздних версиях происходит запуск процесса с низким IL. В Windows XP и Windows Server 2003 новый процесс запускается с помощью маркера, из которого удалено большинство привилегий, а группы с расширенными возможностями помечены флагом **Deny-Only**. Если Procexp работает с правами администратора, у нового процесса будут права, близкие к правам обычного пользователя, а если Procexp работает с привилегиями обычного пользователя, у созданных им процессов будет еще меньше прав.

Сеансы других пользователей

Вкладка **Пользователи** Диспетчера задач отображает сеансы других пользователей, если они открыты на данном компьютере. При наличии прав администратора можно отправить сообщение, которое появится на рабочем столе нужного пользователя, отключить сеанс этого пользователя или принудительно «вывести» его из системы. Соответствующие функции Procexp находятся в меню **Users (Пользователи)**. Также **Procexp** поддерживает окно свойств сеанса, в котором отображается ID сеанса, состояние сеанса, его активность, имя и IP-адрес источника удаленного подключения, а также разрешение дисплея удаленного рабочего стола.

Разные функции

Вот несколько тем, которые, кажется, не раскрыты более нигде.

Выключение ПК

Команда **Shutdown** в меню **File** позволяет выходить из системы, выключать, блокировать и перезагружать компьютер. Возможно также использование параметров **Hibernate (Гибернация)** и **Stand By (Дежурный режим)**, если система поддерживает их.

Переключатели командной строки

В табл. 3-1 описываются параметры командной строки Procexp.

Табл. 3-1. Параметры командной строки

Параметр	Описание
/e	В Vista и выше требует повышения привилегий UAC при запуске Procexp
/x	Запускает Procexp свернутым в области уведомлений («трее»)
/p:r /p:h /p:n /p:l	Устанавливает приоритет процесса для Procexp, соответственно Realtime (Реального времени), High (Высокий), Normal (Обычный) или Low (Низкий). Если приоритет не выбран, то по умолчанию назначается приоритет High
/s:*PID*	Выбирает процесс с идентификатором, заданным в десятичной системе, например Procexp.exe /S:520

Восстановление настроек Procexp по умолчанию

Procexp хранит все свои настройки конфигурации в разделе реестра «HKEY CURRENT USER\ Software\Sysinternals\Process Explorer». Простейший способ восстановления всех настроек конфигурации Procexp — закрыть программу, удалить из реестра ее раздел и снова запустить.

Клавиатурные комбинации

Применяемые в Procexp комбинации клавиш показаны в табл. 3-2.

Табл. 3-2. Клавиатурные комбинации Procexp

Комбинация	Описание
Ctrl+A	Сохранить отображаемые данные в новый файл (эквивалент команды File | Save As)
Ctrl+C	Копировать текущую строку из главного окна или нижней панели
Ctrl+D	Войти в режим просмотра DLL
Ctrl+F	Найти описатель или DLL
Ctrl+H	Войти в режим просмотра описателей
Ctrl+I	Отобразить окно System Information
Ctrl+L	Показать или спрятать нижнюю панель
Ctrl+M	Искать в Интернете
Ctrl+R	Запустить новый процесс (эквивалент команды File | Run)
Ctrl+S	Сохранить показанные данные в файл (эквивалент команды File | Save)
Ctrl+T	Показать дерево процессов (эквивалент команды View | Show Process Tree)
Ctrl+1, Ctrl+2 и т. д.	Загрузить первый, второй и т. д. набор столбцов

Табл. 3-2. (*окончание*)

Комбинация	Описание
Space	Приостановить или возобновить автоматическое обновление
Del	Уничтожить выбранный процесс
Shift+Del	Уничтожить дерево процессов (выбранный процесс с потомками)
Fl	Показать справку
F5	Обновить данные однократно

Глава 4

Process Monitor

Однажды моего соавтора по книге «Windows Internals» («Внутреннее устройство Microsoft Windows», Русская редакция, 2008) Дэвида Соломона пригласили в одну из крупнейших фирм — производителей оборудования, работающего под Windows. Цель — провести мастер-класс по внутреннему устройству Windows для разработчиков программ, функционирующих в режиме ядра. За пару месяцев до мастер-класса представители компании попросили Дэвида использовать на занятиях утилиту-анализатор режима ядра, с которой работают специалисты этой фирмы на ее оборудовании. Дэвид решил, что без труда освоит любую программу этой фирмы, к тому же это позволит ему неплохо заработать, поэтому он согласился.

Разумеется, Дэвид откладывал знакомство с программой до последнего, и запустил ее только в самолете, посмотрев на ноутбуке несколько серий «Звездного пути». Как оказалось, только для того чтобы прочесть сообщение об ошибке: «Данная программа не работает без устройства *той_самой_фирмы*». А в ноутбуке Дэвида этого устройства просто не было, и он пришел в полное замешательство. Неужели завтра утром ему придется признаться, что этот казус обнаружился лишь несколько часов назад?..

Дэвид запаниковал, его прошиб холодный пот, и он попросил стюардессу принести ему выпивки (а точнее, добавки, так как начал он уже во время просмотра «Звездного пути»). Вернувшись спустя несколько минут, стюардесса заметила, что Дэвид сильно не в своей тарелке, и поинтересовалась, не может ли она помочь ему. Приунывший Дэвид, даже не надеясь на то, что она поймет его, кивнул на дисплей и объяснил свою проблему. Она задумалась лишь на секунду и тут же спросила: «А Process Monitor вы пробовали?».

Мораль этой легенды такова: Process Monitor (Procmon) — это главная утилита, которую многие пользователи используют для диагностики проблем с компьютером. Зачастую Procmon является последним средством для устранения неполадок, поскольку в большинстве случаев он точно определяет их причины. Большинство случаев, о которых рассказывают пользователи наших утилит, можно вкратце сформулировать так: «У нас была какая-то непонятная проблема, но мы запустили Procmon и сразу нашли ее причину».

Программа Process Explorer (см. главу 3) представляет собой прекрасный инструмент наблюдения за процессами в системе. Она оказывает нагрузку на ЦП, выделение памяти, DLL, загруженные процессами, используемые системные объекты, контексты безопасности процессов и пр., и пр. Procmon же отображает активность системы с другой точки зрения. Если Procexp, по сути, дает статический «рентгеновский снимок» системы, то Procmon — это «кардиограф», записывающий подробную информацию об активности реестра, файлов, процессов/потоков и сети. Procexp сообщает вам, что тот или иной процесс открыл описатель определенного файла, а Procmon показывает, какие низкоуровневые операции выполняет процесс над этим файлом, когда он это делал, сколько времени потратил, и успешно ли они завершились, а если нет, то почему, а также отображает полный стек вызовов (трассировку — путь к операции) и многое другое.

Поскольку за короткое время могут выполняться миллионы операций, Procmon предоставляет мощные и гибкие возможности фильтрации, так что вы быстро найдете то, что вас интересует. Procmon может работать по сценарию, загружая инструкции из командных файлов с параметрами. Его вывод можно сохранить в файл для дальнейшего просмотра и анализа в другой системе. Иными словами, начинающему пользователю в другом городе не составит особого труда записать трассировку Procmon и отправить ее вам, чтобы вы смогли разрешить его проблему.

Утилита Procmon, созданная в 2006 году, заменила собой два исходных инструмента Sysinternals — Filemon и Regmon. Filemon собирал информацию об активности файловой системы, а Regmon — об активности реестра. Недостатком обоих инструментов было замедление работы при сборе больших объемов информации, а их возможности фильтрации были ограничены. Кроме того, при работе фильтра во время сбора данных невозможно было захватывать отфильтрованные данные, так как они тут же удалялись. Procmon был написан с нуля, в нем реализовано унифицированное представление активности файлов, реестра, процессов, потоков и других сущностей. Новый инструмент способен регистрировать гораздо больше подробностей и эффективнее масштабируется, чем Filemon и Regmon, но при этом куда меньше замедляет работу системы. Кроме того, Procmon поддерживает журнал загрузки, безопасную фильтрацию и форматирование журнала с сохранением всех собранных данных, API-функции для включения в трассировку вывода отладки и многое другое. Так что если вы по привычке все еще пользуетесь Filemon и Regmon, забудьте про них! Filemon и Regmon все еще доступны на сайте Sysinternals для поддержки унаследованных систем, не соответствующих минимальным требованиям для Procmon, но, поскольку тогдашние версии Windows теперь не выпускаются, то и Filemon и Regmon больше не обновляются.

Procmon работает с Windows XP и выше с архитектурой x86 и x64 , а также с Windows Server версии 2003 и выше.

Начало работы с Procmon

Поскольку Procmon загружает драйвер ядра, он требует наличия прав администратора для захвата событий, включая привилегию загружать и выгружать драйверы устройств. В Windows Vista и более поздних версиях Windows автоматически запрашивает повышение прав UAC при запуске Procmon из процесса без повышенных привилегий, например, из Проводника. В Windows XP или Windows Server 2003 необходимо войти в систему в качестве администратора или использовать команду **Запуск от имени** с учетной записью администратора. Более подробную информацию см. в разделе «Права администратора» главы 2.

Примечание Procmon не требует прав администратора, чтобы открыть существующий файл журнала с помощью параметра командной строки */OpenLog*.

Самый простой способ начать работу с Procmon — запустить его. Появится окно Process Monitor, как на рис. 4-1, и сразу начнется вывод данных. Каждая строка таблицы представляет одно низкоуровневое событие, произошедшее в вашей системе. Можно настроить, какие столбцы и в каком порядке будут в таблице, по умолчанию выводятся время, имя и ID процесса, операция (со значком класса, например, операция файловой системы, реестра и т. д.), путь к объекту, в котором выполняется операция (если он есть), ее результат и дополнительные сведения.

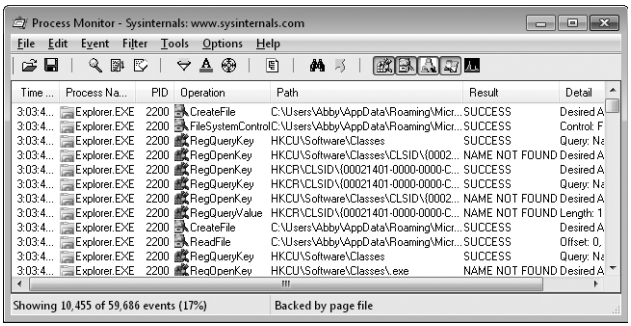


Рис. 4-1. Process Monitor

Строка состояния показывает, помимо прочего, число захваченных событий. Это число будет быстро расти, пока вы не выключите захват событий. Для включения или выключения захвата нажмите клавиши Ctrl + E или щелкните значок **Capture (Захват)** на панели инструментов.

Для удаления всех захваченных событий нажмите клавиши Ctrk+X или щелкните значок **Clear (Очистить)** на панели инструментов.

События добавляются в конец списка в хронологическом порядке. Функция **Auto Scroll (Автопрокрутка)** (по умолчанию отключена) выполняет прокрутку списка по мере добавления событий, так чтобы были видны последние добавления. Для включения и выключения автопрокрутки нажмите клавиши Ctrl+A или щелкните соответствующий значок на панели инструментов.

Настройки отображения Чтобы окно Procmon осталось видимо, даже когда оно не в фокусе, пометьте флажок **Always on Top (Всегда наверху)** в меню **Options (Параметры)**.

Выберите **Font (Шрифт)** в меню **Options** для изменения шрифта, используемого Procmon в главном окне и в других таблицах, например, в диалоговых окнах фильтрации и выделения, во вкладке **Stack (Стек)** параметра **Properties (Свойства)** в меню **Event (Событие)**, а также в диалоговых окнах **Trace Summary (Сводка трассировки).**

События

В табл. 4-1 описываются классы событий, захваченных Procmon.

Табл. 4-1. Классы событий

Значок	Событие	Описание
	Файл реестра (Registry File)	Операции с реестром, в частности, создание, перечисление, запросы, а также удаление параметров и значений
	Система (System)	Операции в локальной памяти и удаленных файловых системах, в том числе, в файловых системах или устройствах, добавленных во время работы Procmon
	Сеть (Network)	Сетевая активность (UDP и TCP), в том числе исходные и конечные адреса (но без переданных или полученных данных). Procmon также поддерживает разрешение адресов в доменные имена и отображение IP-адресов. Параметр Show Resolved Network Addresses (Показать найденные имена сетевых адресов) находится в меню Options. Можно также активировать ее с помощью клавиш Ctrl+N
	Процесс (Process)	События процесса и потоков, в частности, создание процесса родительским процессом, запуск процесса, создание потока, завершение потока или процесса, а также загрузка образов исполняемых файлов и файлов данных в адресное пространство процесса (обратите внимание на то, что Procmon не регистрирует загрузку этих образов)
	Трассировка (Profiling)	Создает и записывает событие для каждого процесса или потока в системе, регистрируя потраченное время в режимах ядра и пользователя, использование памяти и переключения контекстов с момента завершения предыдущей трассировки. По умолчанию включенные в трассировку события для процесса регистрируются всегда, а для потока — никогда. Трассировка вывода отладчика (см. ниже), также входит в данный класс событий

Включение и выключение отображения данных типов событий производится с помощью пяти кнопок в правой части панели инструментов Procmon. Эти кнопки будут описаны далее, в разделе «Фильтрация и выделение».

Совет Событие **Load Image (Загрузка образа)** может помочь в решении проблем с запуском программы. Если программа не запускается, то зачастую найти причину проблемы можно, узнав, какие DLL были загружены последними. Например, в DLL может возникнуть ошибка, вызывающая нарушение доступа. Это может повлечь за собой блокировку загрузчика, «зависание» процесса или невозможность вызова необходимой DLL. В последнем случае за событием **Load Image**, как правило, следуют события файловой системы, соответствующие поиску отсутствующей DLL.

Параметры отображения умолчанию

Procmon отображает данные событий в полях, которые можно настраивать. По умолчанию выводится следующая информация:

- **Время** Время, когда произошло событие (в секундах, отображается с семью десятичными знаками после целого числа, однако фактическая точность зависит от таймера высокого разрешения процессора, точность которого определяется системой). Procmon регистрирует время по Гринвичу (UTC), но отображает время во временной зоне компьютера, на работает программа. Например, если в журнале зафиксировано время 9:00 утра по Восточному времени (UTC5), то в системе, настроенной на Тихоокеанский часовой пояс, будет 6:00 утра.
- **Имя процесса** Имя процесса, выполняющего операцию, вместе со значком из исполняемого файла процесса.
- **ID процесса** Идентификатор процесса.
- **Операция** Имя записанной низкоуровневой операции и значок класса события (активность реестра, файловой системы, сети, процесса или событие трассировки).
- **Путь** Путь к объекту, с которым выполняется операция (если применимо). Возможные пути включают: путь реестра, начинающийся с имени куста, путь файловой системы, начинающийся с буквы диска, UNC-имени, исходных и целевых сетевых адресов и портов. Обратите внимание на то, что HKEY CLASSES ROOT в Win32 — результат объединения HKLM\Software\Classes и HKCU\Software\Classes. Для путей реестра HKCR является синонимом HKLM\Software\Classes; при доступе к разделу из HKCR отображается полный (HKCU или HKU). Кроме того, HKCU является синонимом куста HKEY USERS учетной записи, под которой работает Procmon. Если Procmon работает под записью, отличной от записи процесса, то для данного процесса при обращении к HKCU выводится значение *HW\{SID_пользователя}*.
- **Результат** Результат операции. К основным результатам относятся следующие: *SUCCESS (УСПЕХ), ACCESS DENIED (В ДОСТУПЕ ОТКАЗАНО), NAME NOT FOUND (ИМЯ НЕ НАЙДЕНО), END OF FILE (КОНЕЦ*

ФАЙЛА), а также *BUFFER OVERFLOW (ПЕРЕПОЛНЕНИЕ БУФЕРА)*, последний зачастую понимают неправильно (см. ниже); в табл. 4-2 описаны другие коды результатов.

- **Подробности** Дополнительная информация о событии, относящаяся к операции, в частности, желаемый уровень доступа при первом открытии объекта, размер, тип и содержание данных при чтении параметра реестра, число битов данных при отправке и получении по сети. Некоторые операции файловой системы содержат коды атрибутов файлов (табл. 4-3).

Переполнение буфера По мере учащения атак через Интернет термин «переполнение буфера» стал синонимом захвата вредоносными программами обеспечением несанкционированного контроля над удаленным компьютером. В этом контексте переполнение буфера происходит, когда программа копирует в буфер запоминающего устройства больше данных, чем он вмещает, что приводит к перезаписи кода программы и кодом, нужным злоумышленнику, который потом исполняется. Неудивительно, что начинающие пользователи Procmon пугаются надписи «*BUFFER OVERFLOW*» в поле «Результат», хотя оснований для беспокойства нет.

Код *STATUS BUFFER OVERFLOW* (*ПЕРЕПОЛНЕНИЕ БУФЕРА СОСТОЯНИЯ)*, как результат *NTSTATUS* регистрируется, когда программа запрашивает значение переменной длины, например, параметр реестра, но не предоставляет буфер достаточного размера для получения этой информации, так как «не знает» размер запрошенного значения заранее. Система сообщает программе необходимый размер буфера и копирует в буфер столько данных, сколько может, но фактически буфер не переполняется. Типичный прием при программировании таков: после получения *BUFFER OVERFLOW* программа выделяет буфер достаточного размера и снова запрашивает те же данные, в итоге операция завершается с кодом *SUCCESS*.

Табл. 4-2. Основные коды результатов

Результат	Описание
SUCCESS (УСПЕХ)	Операция успешно завершена
ACCESS DENIED (ОТКАЗАНО В ДОСТУПЕ)	Операция не выполнена, так как дескриптор защиты объекта не дает программе необходимые права или целевой файл доступен только для чтения (редко знаменует фатальные события при отладке)
SHARING VIOLATION (НАРУШЕНИЕ СОВМЕСТНОГО ДОСТУПА К ФАЙЛУ)	Операция не была выполнена, поскольку объект уже открыт и не поддерживает режим совместного использования ресурсов, запрашиваемый программой
NAME COLLISION (КОНФЛИКТ ИМЕН)	Программа пыталась создать уже существующий объект
NAME NOT FOUND (ИМЯ ФАЙЛА НЕ НАЙДЕНО)	Программа пыталась открыть несуществующий объект. Возврат такого кода возможен, когда загрузчик ищет DLL в разных директориях
PATH NOT FOUND (ПУТЬ НЕ НАЙДЕН)	
NO SUCH FILE (ФАЙЛ С ТАКИМ ИМЕНЕМ НЕ СУЩЕСТВУЕТ)	

Табл. 4-2. (*продолжение*)

Результат	Описание
NAME INVALID (НЕПРАВИЛЬНОЕ ИМЯ)	Программа запросила объект с неправильным именем, например, *C:\Windows\»regedit.exe»*
NO MORE ENTRIES (ЗАПИСИ ОТСУТСТВУЮТ)	Программа завершила перечисление содержимого папки или раздела реестра
NO MORE FILES (ФАЙЛЫ ОТСУТСТВУЮТ)	
END OF FILE (КОНЕЦ ФАЙЛА)	Программа прочитала файл до конца
BUFFER TOO SMALL (НЕДОСТАТОЧНЫЙ ОБЪЕМ БУФЕРА)	По сути то же, что и BUFFER OVERFLOW. При устранении неполадок, как правило, не несет существенной информации
REPARSE (ПОВТОРНЫЙ РАЗБОР)	Программа запросила объект, который ссылается на другой объект. Например, HKLM\System\CurrentControlSet может перенаправить на HKLM\System\ControlSet001
NOT REPARSE POINT (НЕТ ТОЧКИ ПОВТОРНОГО РАЗБОРА)	Запрашиваемый объект не ссылается на другой объект
FAST I0 DISALLOWED (БЫСТРЫЙ ВВОД-ВЫВОД ЗАПРЕЩЕН)	Низкоуровневый оптимизированный механизм недоступен для запрашиваемого объекта файловой системы. Редко бывает важным при устранении неполадок
FILE LOCKED WITH ONLY READERS (ФАЙЛ ДОСТУПЕН ТОЛЬКО ДЛЯ ЧТЕНИЯ)	Файл или проекция файла заблокирована и доступна только для чтения
FILE LOCKED WITH WRITERS (ФАЙЛ ДОСТУПЕН ДЛЯ ЗАПИСИ)	Файл или проекция файла заблокирована, и минимум один пользователь может записывать в него данные
IS DIRECTORY (ОБЪЕКТ ЯВЛЯЕТСЯ КАТАЛОГОМ)	Запрашиваемый объект является каталогом файловой системы
INVALID DEVICE REQUEST (НЕДОПУСТИМЫЙ ЗАПРОС УСТРОЙСТВА)	Данный запрос не является допустимой операцией для целевого устройства
INVALID PARAMETER (НЕВЕРНЫЙ ПАРАМЕТР)	Службе или функции был передан неверный параметр
NOT GRANTED (НЕ ПРЕДОСТАВЛЕНО)	Запрашиваемая блокировка файла не может быть предоставлена из-за наличия других блокировок
CANCELLED (ОТМЕНЕНО)	Запрос ввода-вывода, например, проверка каталога файловой системы на наличие изменений, был отменен

Табл. 4-2. (*продолжение*)

Результат	Описание
BAD NETWORK PATH (СЕТЕВОЙ ПУТЬ НЕ НАЙДЕН)	Сетевой путь не удается найти
BAD NETWORK NAME (СЕТЕВОЕ ИМЯ НЕ НАЙДЕНО)	Указанное сетевое имя не может быть найдено на удаленном сервере
MEDIA WRITE PROTECTED (НОСИТЕЛЬ ЗАЩИЩЕН ОТ ЗАПИСИ)	Носитель не может быть записан, так как защищен от записи
KEY DELETED (РАЗДЕЛ УДАЛЕН)	Попытка несанкционированной операции с разделом реестра, который был подготовлен к удалению
NOT IMPLEMENTED (НЕ РЕАЛИЗОВАНА)	Запрашиваемая операция не выполнена, т. к. не реализована в целевом объекте

Настройка отображения полей

Часто информация, содержащаяся в поле, слишком велика и не отображается целиком. В этом случае можно навести на него указатель мыши, и полный текст содержимого поля появится в виде всплывающей подсказки. Размеры полей можно изменять, перетаскивая их границы в заголовках. Можно установить автоматический подбор ширины полей под содержимое двойным щелчком границы справа от названия поля. Чтобы изменить порядок полей, перетащив их заголовки как требуется.

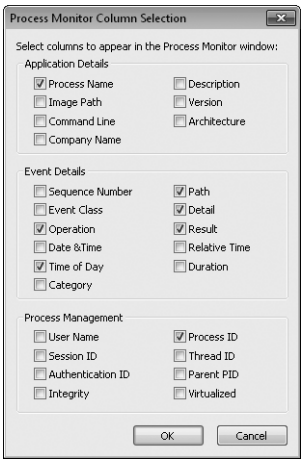


Рис. 4-2. Диалог Column Selection

Можно включать и выключать вывод отдельных полей, щелкнув правой кнопкой мыши «шапку» списка и выбрав пункт **Select Columns (Выбрать поля)** либо вызвав команду **Select Columns** из меню **Options**. Как показано

на рис. 4-2, поля сгруппированы по разделам: **Application Details (Сведения о приложениях)**, **Event Details (Сведения о событиях)** и **Process Management (Управление процессами)**.

Раздел **Application Details** содержит статическую информацию, которая определяется при запуске процесса и не изменяется до его завершения, например путь к образу, командная строка и тип архитектуры.

Раздел **Event Details** содержит информацию о конкретном событии. Помимо полей по умолчанию, выводятся:

- **Sequence Number** на текущей странице.
- **Event Class:** реестр, файловая система, сеть, процесс, трассировка.
- **Category** (если есть): чтение, запись, чтение и запись метаданных.
- **Relative Time** прошедшее от запуска Procmon или последней очистки экрана до операции.
- **Duration** операции в секундах. Для событий трассировки потоков это сумма времени ядра и времени пользователя, затраченных потоком с момента предыдущего события трассировки. Для событий трассировки процесса эта величина установлена в ноль (см. ниже).

Раздел **Process Management** содержит следующую информацию о процессе:

- **User Name** Участник системы безопасности, под учетной записью которого исполняется процесс.
- **Session ID** Сеанс служб терминалов, в котором работает процесс. Службы всегда работают в сеансе 0 (см. главу 2).
- **Integrity** Уровень целостности (IL) процесса, выполняющего операцию (только в Windows Vista и выше).
- **Thread ID** Идентификатор потока, выполняющего операцию, отображается как «TID».
- **Virtualized** Показывает, включена ли виртуализация UAC для процесса, выполняющего операцию (только в Windows Vista и выше). Не путайте эту виртуализацию с виртуализацией приложений и компьютеров.

Диалог Event Properties

Для получения более подробной информации о событии дважды щелкните запись события — откроется окно **Event Properties (Свойства события)**. При нажатии Ctrl+K это окно открывается с вкладкой **Stack**. Это окно — не модальное, поэтому можно работать с главным окном Procmon, не закрывая это окно, а также открывать сразу несколько таких окон. Можно также изменять размеры окон и раскрывать их на весь экран.

Кнопки со стрелками (рис. 4-3) позволяют прокручивать список событий. Если вы установите флажок **Next Highlighted (Следующее выделенное)**, щелчок кнопки со стрелкой откроет следующее выделенное (см. ниже) событие.

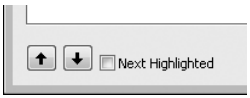


Рис. 4-3. Кнопки навигации в диалоговом окне Event Properties

Щелчок кнопки **Copy All (Копировать все)** приводит к копированию содержимого текущей вкладки в буфер обмена как обычного текста.

Вкладка Event

Вкладка **Event** в диалоговом окне **Event Properties** (рис. 4-4) отображает следующую информацию о событии: дату и время, идентификатор потока, класс события, операцию, результат, путь и длительность. Под горизонтальной строкой находится информация об отдельных операциях. Та же информация есть и в поле **Details**, однако здесь она представлена в более удобной форме.

Табл. 4-3. Атрибуты файлов в поле Details

Атрибут	Значение
A	Архивный файл или каталог. Приложения обычно используют данный атрибут для резервирования или удаления файлов
C	Сжатый файл или каталог. Содержимое файлов сжато, новые файлы и вложенные каталоги в таких каталогах сжимаются по умолчанию
D	Объект является каталогом или устройством
E	Зашифрованный файл или каталог. Потоки файлов зашифрованы, новые файлы и вложенные каталоги в таких каталогах шифруются по умолчанию
H	Скрытый файл или каталог. Не включен в стандартный список каталогов
N	Файл, не имеющий других атрибутов. Используется только в одиночку
NCI	Файл или каталог не индексируется службой индексирования
0	Данные файла в настоящее время недоступны (файл перенесен в автономное хранилище). Используется программой Remote Storage
R	Файл доступен только для чтения. Приложения могут читать его, но не могут записывать или удалять. Этот атрибут игнорируется для каталога
RP	Файл или каталог, с которым связана точка повторного разбора либо файл с символической связью
S	Файл или каталог, который сообща либо монопольно используется операционной системой
SF	Разреженный файл
T	Временный файл. Файловые системы не сбрасывают данные на накопитель при наличии достаточного объема кеш-памяти, т.к. приложение обычно удаляет временный файл после закрытия его описателя. При использовании временного файла система может обойтись вовсе без записи, в противном случае после закрытия описателя файла данные записываются

На рис. 4-4 изображена попытка выполнения операции CreateFile в корневом каталоге диска C, результат которой — Access Denied. Подробности содержат требуемый доступ. **Строка Disposition (Расположение)** показывает, что в случае успешного завершения операции был бы открыт существующий объект, а не создан новый. Строка **ShareMode** показывает, что доступ не является эксклюзивным, и другие процессы могут открывать объект для чтения, записей и удаления. Очевидно, что эти детали относятся к операции CreateFile и не имеют отношения, например, к операции Load Image (Если ширина текста слишком большая для дисплея, можно уменьшить размер шрифта или увеличить диалоговое окно. Можно также выбрать *Копировать все* либо щелкнуть правой кнопкой мыши диалоговое окно **Details**, выделить все и копировать, а затем вставить текст в другом месте).

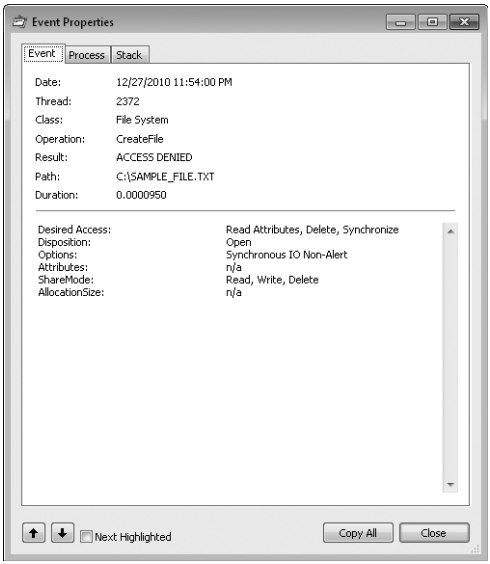


Рис. 4-4. Вкладка Event в диалоговом окне Event Properties

Вкладка Process

Вкладка **Process** диалогового окна **Event Properties** (рис. 4-5) отображает подробную информацию о генерации выбранного события на момент его возникновения.

Вкладка **Process** содержит следующую информацию:

- Значок приложения, извлеченный из образа процесса (или значок по умолчанию, если у образа его нет).
- Описание, имя компании и версия файла, извлеченные из информационного ресурса образа.
- Имя процесса.
- Путь к файлу исполняемого образа.
- Командная строка, которая использовалась для запуска данного процесса.

- ID процесса и ID процесса-предка, запустившего его.
- ID сеанса служб терминалов, в котором работает данный процесс.
- Учетная запись пользователя, под которой работает процесс.
- Идентификатор аутентификации (Auth ID) для маркера процесса. Идентификатор аутентификации — это локально уникальный идентификатор (LUID), идентифицирующий сеанс LSA, создавший маркер доступа для данного процесса. (LUID — это сгенерированное системой 64-разрядное значение, гарантированно уникальное в течение сеанса, в котором оно было сгенерировано). Подробнее об утилите LogonSessions, регистрирующей активные сеансы входа LSA, см. в главе 8.
- Время начала и время завершения процесса (если оно зарегистрировано).
- Архитектура (32- или 64-разрядная).
- Активированы ли UAC и виртуализация реестра для данного процесса (только в Windows Vista и выше).
- Уровень целостности процесса (только в Windows Vista и более поздних версиях).
- Список модулей (исполняемых образов), загруженных в адресное пространство процесса на момент данного события. В недавно запущенном процессе список будет пустым до тех пор, пока какие-либо события Load Image не загрузят exe-файл, Ntdll.dll или другие модули.

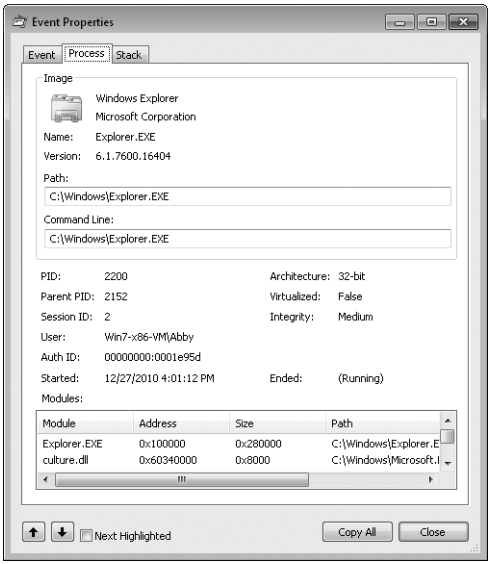


Рис. 4-5. Вкладка Process в диалоговом окне Event Properties

Вкладка Stack

Вкладка **Stack** диалогового окна **Event Properties** (рис. 4-6) показывает стек вызовов потока в момент регистрации события. По стеку можно определить

причину события и отвечающего за него компонента. Чтобы понять, что такое стек вызовов и как настроить Procmon для максимального отображения информации, которую можно от него получить, см. раздел «Стеки вызовов и символы» Главы 2.

Каждая строка представляет один фрейм стека и содержит пять полей:

- **Frame** Отображает номер фрейма, а также букву K для фрейма в режиме ядра или U для фрейма режима пользователя (фреймы стека в пользовательском режиме не записываются в 64-разрядных версиях Windows до Windows Vista SP1 и Windows Server 2008).
- **Module** Имя файла, содержащего код, исполняемый в данном фрейме.
- **Location** Адрес кода внутри модуля, где исполняется код. Если имеются символы, указывается имя функции и смещение от начала данной функции. Если есть и исходный код, выводится путь к нему и номер строки. Если символы недоступны, но модуль имеет таблицу экспорта, выводится ближайшая предшествующая экспортированная функция со смещением относительно нее. Если ни символы, ни экспортированные функции не доступны, выводится смещение от базового адреса модуля в памяти (см. главу 2).
- **Address** Адрес команды в виртуальном адресном пространстве исполняемого процесса.
- **Path** Полный путь к файлу, указанному в поле **Module**. Если окно имеет размер по умолчанию, придется прокрутить или изменить его размер, чтобы увидеть это поле, по которому можно определить, какая версия DLL исполняется.

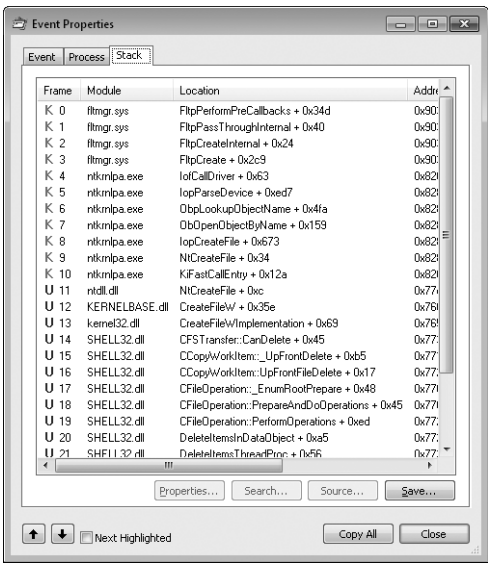


Рис. 4-6. Вкладка Stack в диалоговом окне Event Properties

На вкладке **Stack** вы можете выполнять следующие операции:

- щелкнуть кнопку **Save (Сохранить)** для сохранения трассировки стека в CSV-файл;
- дважды щелкнуть строку в трассировке стека, чтобы открыть диалоговое окно **Module Properties (Свойства модуля).** Оно отображает имя и путь модуля, а также описание, версию файла и имя компании, извлеченные из данных о версии модуля;
- выбрать строку и щелкнуть **Search (Поиск)** для поиска в Интернете более подробной информации о символах или модуле, указанном в поле **Location**. Procmon инициирует поиск с помощью браузера и поисковой системы по умолчанию;
- щелкнуть кнопку **Source (Источник),** которая активируется, если символьная информация о выбранном фрейме стека включает сведения об исходном файле. Исходный файл (если он найден в предполагаемом расположении) отображается в новом окне, при этом выбирается идентифицированная строка исходного кода.

Примечание Чтобы Procmon активировал некоторые из этих функций, должны быть подключены символы. Для этого в окне Procmon (см. рис. 4-1) щелкните **Configure Symbols (Настроить символы)** в меню **Options**, подробнее см. в главе 2.

Отображение событий трассировки

Четыре класса событий, отображаемых Procmon по умолчанию, охватывают активность реестра, файловой системы, сети и процессов. Эти события генерируются процессами, работающими на компьютере. Пятый класс событий — события трассировки — «искусственные» события, периодически генерируемые самой программой Procmon (за исключением событий трассировки вывода отладчика, описанных ниже). События трассировки по умолчанию не отображаются, но можно включить их вывод с помощью кнопки **Show Profiling Events (Показывать события трассировки)** на панели инструментов.

События трассировки процесса ежесекундно генерируются на компьютере для каждого процесса. С каждым событием регистрируются: время ЦП в режимах пользователя и ядра с момента запуска процесса, а также число байтов закрытой памяти, выделенной процессу на данный момент и размер рабочего набора. Атрибут **Duration (Длительность)** при трассировке процесса установлен в ноль.

В отличие от событий трассировки процесса, данные, записываемые событиями трассировки потока, не накапливаются. Активные события трассировки регистрируют время ЦП в режимах пользователя и ядра, а также число переключений контекста с момента предыдущей трассировки потока. Атрибут **Duration** сообщает общее время ЦП в режимах пользователя и ядра, он может использоваться в правилах фильтрации для выявления пиковых нагрузок на ЦП. События трассировки потока создаются только

для потоков, имеющих не менее одного переключения контекста во время интервала опроса, но не для потоков неработающего процесса.

События трассировки процесса всегда генерируются один раз в секунду. События трассировки потока не генерируются по умолчанию, но их генерацию можно включить в окне **Thread Profiling Options (Параметры трассировки потока)** (рис. 4-7), которое открывается при выборе параметра **Profiling Events (События трассировки)** в меню **Options**. При установке флажка **Generate Thread Profiling Events (Генерировать события трассировки потока)** Procmon генерирует события трассировки потока раз в секунду либо в 100 миллисекунд в зависимости от настроек в окне **Options**.

Внимание! Трассировка потоков требует больших затрат системных ресурсов, поэтому не включайте ее без необходимости.

Рис. 4-7. Диалоговое окно Thread Profiling Options

Поиск события

Чтобы найти событие в главном окне Procmon по тексту в записи события, откройте окно **Find (Найти)**. Для этого нужно нажать клавиши Ctrl + F или щелкнуть кнопку с изображением бинокля на панели инструментов. Введите текст, который вы хотите найти, и щелкните **Find Next (Найти далее)**. Procmon выберет следующее событие, содержащее искомый текст в любом из отображаемых полей. Чтобы повторить поиск и найти следующее подходящее событие, нажмите клавишу F3. Функция **Find** удобна для быстрого поиска событий, при этом виден контекст предыдущих и последующих событий, но он может быть скрыт, если использовалась фильтрация (см. ниже).

Копирование данных события

Нажмите клавиши Ctrl+C, чтобы скопировать запись выбранного события в буфер обмена в виде текста с разделителями-табуляторами. Имейте в виду, что для выбора нескольких объектов в списке работают стандартные методы Windows, в том числе Shift + клавиша–стрелка или Shift + щелчок для выбора дополнительных смежных элементов, а также Ctrl + щелчок для выбора несмежных элементов списка. Procmon послушно скопирует текст из любых полей выбранных записей.

Можно также скопировать текст из отдельного поля, щелкнув правой кнопкой мыши поле и выбрав **Copy** в контекстном меню. В примере, показанном на рис. 4-8, текст «HKCR\.exe\OpenWithProgids» копируется в буфер обмена путем выбора восьмого элемента в контекстном меню.

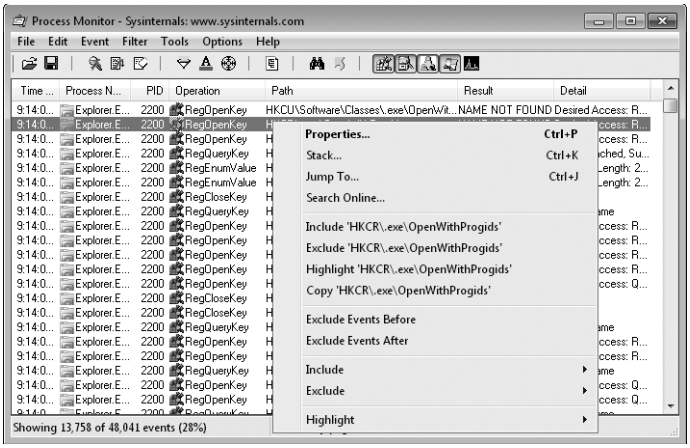


Рис. 4-8. Контекстное меню поля Path

Переход к параметру реестра или файлу

Чтобы перейти к нужному разделу реестра или файлу, выберите событие реестра или файловой системы с указанным путем и нажмите Ctr+J. Procmon запустит Regedit (в случае реестра) или откроет новое окно **Проводника** (для файловой системы) и перейдет по выбранному пути. Переход можно также выполнить щелчком значка **Jump To Object (Перейти к объекту)** на панели инструментов или выбрав **Jump To (Перейти)** в контекстном меню события, как показано на рис. 4-8.

Поиск в Интернете

Возможен поиск имени процесса или события в Интернете. Для этого нужно выбрать событие и щелкнуть **Search Online (Поиск в Интернете)** в меню **Event**, как показано на рис. 4-8. Procmon начнет поиск с помощью вашего браузера и поисковой системы по умолчанию. Это удобно при поиске вредоносного ПО или программ, запускающих неизвестные процессы.

Фильтрация и подсветка

Procmon способен без труда захватывать за короткое время миллионы событий от десятков процессов. Чтобы отобрать интересующие вас события, используйте мощные и гибкие параметры фильтрации Procmon, позволяющие ограничивать количество отображаемых элементов. Те же функции применяются и для поиска конкретных событий. В примере, показанном на рис. 4-9, Procmon отображает только результаты ACCESS DENIED в

Cinmania.exe и выделяет те события, в которых путь начинается с “C:\Windows\Fonts”. Строка состояния показывает, что из 355859 событий, содержащихся в журнале, только 63 соответствуют критериям фильтра и отображаются на дисплее, а больше 99,9% записанных событий скрыто.

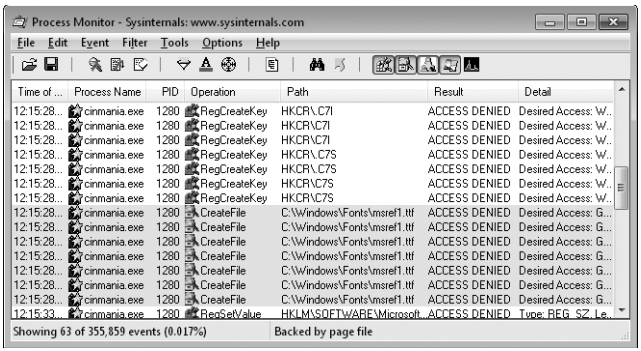


Рис. 4-9. Пример фильтрации и выделения в Procmon

Возможности для фильтрации у Regmon и Filemon были ограниченными. Одним из наиболее существенных ограничений было необратимое удаление элементов во время применения фильтра. В Procmon же отфильтрованные элементы только скрываются, но не выкидываются из исходных данных. Их можно вернуть на экран, изменив или удалив фильтр.

Настройка фильтров

Вы можете настраивать фильтры, используя любые атрибуты событий, как отображаемые, так и скрытые. Можно искать точное совпадение указанного значения либо частичные совпадения, например по условию «begins with» («начинается с»), «ends with» («заканчивается на»), «contains» («содержит»), «less than» («меньше чем») или «more than» («больше чем»), об атрибутах см. выше.

Простейшим фильтрами являются фильтры по классу события, представленные пятью кнопками в правой части панели инструментов (см. рис. 4-10), включающие отображение событий реестра, файловой системы, сети, процессов (потоков) и трассировки. Когда кнопка класса событий отключается, добавляется соответствующий фильтр **Exclude (Исключить)**, скрывающий все события данного класса.

Рис. 4-10. Кнопки фильтров по классу событий на панели инструментов Procmon

Еще один простой способ настройки фильтра — использование команды **Include Process From Window (Включить процесс из окна)**. Эта функция создает фильтр по идентификатору процесса, которому соответствует открытое окно. Выберите кнопку с изображением прицела на панели инструментов и перетащите ее в нужное окно. Окно Procmon во время этой

операции не отображается, а вокруг окна, над которым находится указатель мыши, выводится рамка. Отпустите кнопку мыши, и Procmon вновь появится, уже с добавленным в фильтр идентификатором процесса, которому принадлежит выбранное окно.

Полный набор параметров фильтрации см. в окне **Filter (Фильтр)** (рис. 4-11), открывается оно нажатием Ctrl+L или кнопкой **Filter** на панели инструментов. Заметьте, что в фильтре по умолчанию уже есть правила **Exclude** (о них будет сказано ниже).

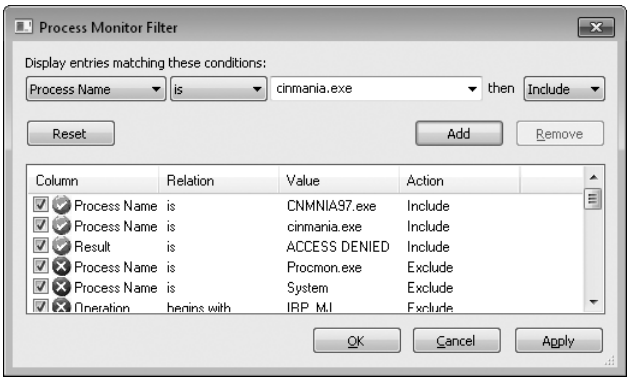


Рис. 4-11. Диалоговое окно Filter

Для добавления правила фильтра выберите атрибут из первого раскрывающегося списка, тип проверки из второго и величину для сравнения — из третьего. Все операции сравнения текстовых значений чувствительны к регистру. Когда вы выбираете атрибут из первого списка, список в третьем поле заполняется возможными вариантами. Например, если выбрать **Process Name (Имя процесса)**, в список третьего поля будут внесены имена процессов, сгенерировавших события (только для атрибутов вроде **Path** это не делается, т.к. список возможных вариантов может быть слишком длинным). Можно также отредактировать значение прямо в поле со списком. В первой строке четвертого поля со списком можно добавлять и удалять отображаемые события. Щелкните кнопку **Add (Добавить)**, чтобы добавить новые параметры фильтрации к существующему фильтру. Закончив настройку списка фильтров, щелкните **OK** или **Apply (Применить)**.

Для редактирования или удаления правила из фильтра дважды щелкните его либо выделите и щелкните кнопку **Remove (Удалить)**. Оно будет удалено из списка и возвращено в список правил, откуда его легко вернуть обратно, при необходимости сначала отредактировав. Можно просто отключать отдельные правила, не удаляя их — достаточно снять соответствующий флажок. Чтобы снова активировать правило, установите флажок снова и щелкните **OK** или **Apply**.

Для восстановления значений фильтра по умолчанию щелкните кнопку **Reset (Сброс)** в диалоговом окне **Filter**. Чтобы выполнить данную операцию из главного окна, нужно нажать клавиши Ctrl + R.

Procmon может устанавливать все правила фильтра для одного атрибута, а также добавлять один и тот же фильтр для различных атрибутов. Например, если выбрать **Process Name**, **Include** для значений *Notepad.exe* и *Cmd.exe*, а для элемента **Path** — команду **Include** с значением *C:\Windows,* Procmon отобразит только события, связанные с путем *C:\Windows,* сгенерированные блокнотом и командной строкой, прочие события отображаться не будут.

Вот еще один эффективный способ добавления критериев фильтра: щелкните событие правой кнопкой мыши и выберите критерии из контекстного меню. На рис. 4-12 изображено только контекстное меню из рис. 4-8 с доступными вариантами.

В контекстном меню доступны «быстрые фильтры» для текущего параметра. Например, у четвертого и пятого объектов на рис. 4-12 отображаются «быстрые фильтры» **Include** и **Exclude** для пути реестра HKCR\.exe\OpenWithProgids. Параметр **Exclude Events Before (Исключить события до)** скрывает все события, предшествующие выбранному, добавляя правило, основанное на атрибуте события **Date & Time (Дата и время)**. Аналогичным образом параметр **Exclude Events After (Исключить события после)** скрывает все события, следующие за выбранным. Наконец, подменю **Include** и **Exclude** (см. второй и третий объекты снизу) содержат наиболее востребованные атрибуты фильтров. Выберите имя атрибута в одном из этих подменю, и к текущему фильтру добавится соответствующее значение. Также можно создать фильтр по группе одновременно выбранных событий. Для этого нужно щелкнуть правой кнопкой мыши события и выбрать имя атрибута из подменю **Include** или **Exclude**. В результате будет создан фильтр, выделяющий уникальные значения, содержащиеся в выбранных событиях.

Рис. 4-12. Контекстное меню из рис. 4-8

Эти способы настройки фильтра можно комбинировать. Допустим, нужно отфильтровать процессы, имеющие доступ к разделу реестра HKCR\CLSID\{DFEAF541-F3E1-4C24-ACAC-99C30715084A} и его подразделам. Для начала выберите фильтр «begins with». Чтобы обойтись без длительного ввода с клавиатуры и нудных операций копирования-вставки, найди-

те событие для данного раздела, щелкните его правой кнопкой и выберите **Include** HKCR\CLSID\{DFEAF541-F3E1-4C24-ACAC-99C30715084A} — будет создан фильтр по значению **Path**. Теперь нажмите Ctrl + L, чтобы открыть диалоговое окно **Filter**, дважды щелкните новые критерии, чтобы переместить их из списка в список правил, замените «is» на «begins with», при необходимости отредактируйте путь, щелкните **Add**, затем **ОК**.

Диалоговые окна **Process Tree (Дерево процессов)** и **Summary (Сводка)**, описываемые далее в данной главе, также поддерживают настройку фильтров.

Procmon запоминает последний установленный фильтр и открывает окно **Filter** перед началом захвата событий при следующем запуске. Вы сможете сохранить, отредактировать или изменить фильтр перед захватом данных. Этот шаг можно пропустить, запустив Procmon с параметром командной строки */Quiet*. Можно автоматически очистить фильтр при запуске с помощью параметра командной строки */NoFilter* (подробнее об этом — ниже).

Настройка подсветки

Если фильтрация скрывает лишние события, то подсветка подчеркивает нужные события в списке. По умолчанию события подсвечиваются ярко-синим фоном. Можно изменять цвета текста и фона, выбрав **Highlight Colors (Цвета подсветки)** в меню **Options**.

Настройка подсветки практически аналогична настройке фильтров. Чтобы открыть диалоговое окно **Highlight (Подсветка)**, нужно нажать клавиши Ctrl + H или щелкнуть значок **Highlight** на панели инструментов. Диалоговое окно **Highlight** работает точно так же, как и диалоговое окно **Filter**. Контекстное меню выбранных событий содержит те же параметры, что и для фильтров.

Диалоговое окно **Event Properties**, описанное ранее в этой главе, позволит вам увидеть следующий или предыдущий объект в списке событий. Установив флажок **Next Highlighted**, вы можете перейти к следующему или предыдущему событию.

Расширенный вывод

По умолчанию Procmon скрывает события, не важные для устранения неполадок приложений:

- события, генерируемые самим Procmon;
- события, генерируемые системными процессами;
- события трассировки;
- низкоуровневые операции, имена которых начинаются с *IRP MJ* (пакеты запросов ввода-вывода, используемые драйверами Windows, устройств ввода-вывода, устройств PnP и в прочих функциях ввода-вывода);
- низкоуровневые операции, имена которых начинаются с *FASTIO*. Они аналогичны пакетам запросов ввода-вывода, но, в отличие от них, ис-

пользуются системой ввода-вывода и используют драйвер файловой системы или диспетчер кеша для завершения запроса ввода-вывода;
- результаты, начинающиеся с *FASTI0*, например, *FASTI0 DISALLOWED*;
- деятельность, связанная с системным файлом подкачки;
- активность диспетчера NTFS и MFT.

Установка флажка **Advanced Output (Расширенный вывод)** в меню **Filter** снимает все эти исключения (кроме событий трассировки) и отображает имена уровня драйвера для операций файловой системы. Например, операция, отображаемая как CreateFile в основном режиме, в расширенном режиме отображается как IRP MJ CREATE. Если снять флажок **Advanced Output**, все имена вернутся к исходному виду.

При установке флажка **Advanced Output** функция **Reset Filter** удаляет все правила фильтра, кроме правила, исключающего события трассировки.

Чтобы отслеживать всю деятельность системы, сохранив стандартные имена событий, удалите фильтры по умолчанию, не включая расширенный вывод.

Сохранение фильтров

Созданный фильтр можно сохранить. Это позволит вам быстро загружать и применять сложные фильтры, а также легко переключаться с одного набора фильтров на другой. Кроме того, можно экспортировать сохраненные фильтры и импортировать их в другую систему или другую учетную запись пользователя.

Для сохранения фильтра выберите **Filter | Save Filter** и введите его имя, как показано на рис. 4-13. Procmon предлагает по умолчанию имена Filter 0, Filter 1 и т. д. Можно также выбрать более информативное имя, например, «Операции записей IE».

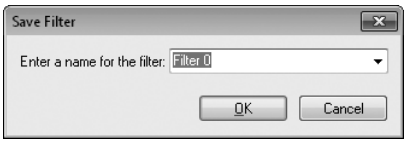


Рис. 4-13. Диалоговое окно Save Filter

Для загрузки и применения сохраненного фильтра выберите его в меню **Filter | Load Filter.** Фильтры располагаются в меню в алфавитном порядке (см. рис. 4-14).

Фильтры можно переименовывать и удалять с помощью диалогового окна **Organize Filters (Управление фильтрами),** как показано на рис. 4-15. Выберите **Organize Filters** в меню **Filter.** Чтобы экспортировать фильтр, выберите его из списка, щелкните кнопку **Export (Экспорт)** и укажите расположение файла. Procmon использует расширение *.PMF для идентификации файлов своих фильтров. Чтобы импортировать фильтр щелкните **Import (Импорт)** и выберите ранее экспортированный фильтр.

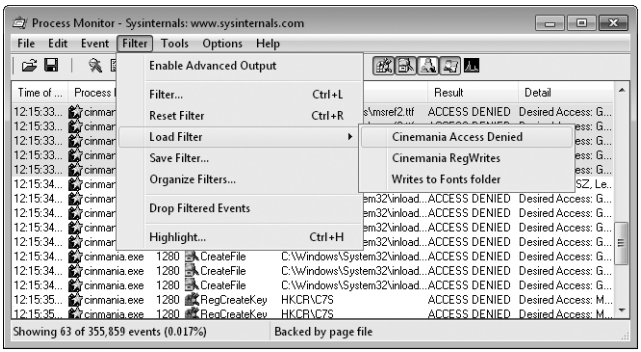


Рис. 4-14. Меню Load Filter

Обратите внимание на то, что сохраненные и экспортируемые фильтры записывают только правила фильтров. Правила подсветки можно сохранять только путем экспорта конфигурации Procmon (включающей также правила фильтров), более подробно об этом — ниже.

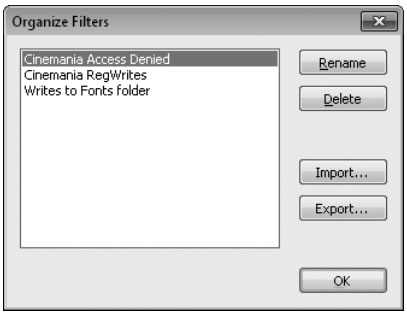


Рис. 4-15. Диалоговое окно Organize Filters

Дерево процессов

Если нажать клавиши Ctrl+T или щелкнуть кнопку **Process Tree (Дерево процесса)** на панели инструментов, откроется окно **Process Tree** (рис. 4-16). Оно отображает в виде дерева все процессы, фигурирующие в загруженной трассировке, как в Procexp. Элементы дерева процессов можно сворачивать и разворачивать, щелкая значки со знаком (+) или (–) слева от процессов-предков, либо выбирая узлы и нажимая клавиши правой и левой стрелок. Предки процессов, расположенных по левой стороне, не генерировали события в трассировке.

Каждое имя процесса находится рядом с соответствующим значком приложения. Значок недоступен, если процесс завершился во время трассировки. Для отображения только процессов, работавших в конце текущей трассировки, установите соответствующий флажок в верхней части диалогового окна.

Строка в нижней части окна отображает информацию о процессе, включая его идентификатор, описание, путь, командную строку, время начала и

завершения (если имеется), имя компании и учетную запись, под которой работает процесс. Эта информация отображается и в самой таблице, вместе с графическим представлением временной шкалы работы процесса.

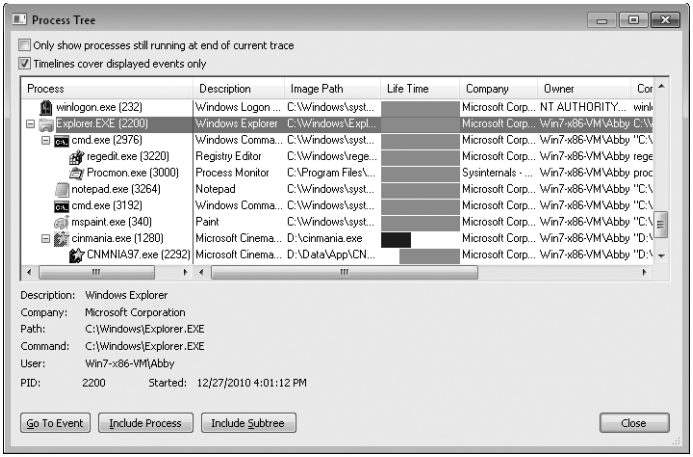


Рис. 4-16. Диалоговое окно Process Tree

Колонка **Life Time (Время жизни)** отображает временную шкалу процесса относительно трассировки или сеанса, в зависимости от того, выбран ли параметр **Timelines Cover Displayed Events Only (Шкала только для отображаемых событий)**. Если данный параметр выбран, выводится зеленая полоса от одного края столбца до другого, свидетельствующая, что процесс уже работал в момент начала трассировки и продолжил работать после ее завершения. Зеленая строка, продолжающаяся справа (см., например, последний элемент дерева на рис. 4-16), показывает относительное время работы процесса после начала трассировки. Более темная зеленая полоса показывает процесс, завершившийся во время трассировки, по ее длине можно судить о моменте его завершения. Если же параметр **Timelines Cover Displayed Events Only** не выбран, времена жизни процессов отсчитываются относительно сеанса работы системы: зеленая полоса у левого края поля показывает процесс, работавший с момента запуска системы или запущенный вскоре после него.

Помимо графического изображения «родства» процессов, в т. ч. завершенных, **Process Tree** помогает выявлять нестандартные события, например процессы, которые постоянно создаются и тут же завершаются.

Если выбрать процесс из дерева и щелкнуть кнопку **Include Process,** то к фильтру с выбранным ID процесса будет добавлено правило «PID is». Если щелкнуть кнопку **Include Subtree (Включить ветвь дерева),** то для выбранного процесса и всех его потомков будет добавлено правило фильтра «PID is».

Чтобы найти событие в трассировке, выберите процесс в дереве и щелкните **Go To Event (Перейти к событию)** — procmon найдет и выберет первое событие в трассировке, *видимое* в главном окне. Обратите внимание:

фильтры могут скрывать все события процесса. Но даже процесс, не выполнивший никакого кода во время трассировки, может отображаться в дереве благодаря событиям трассировки (если последние не скрыты фильтром).

Запись и загрузка трассировки Procmon

«Пожалуйста, пришлите журнал Procmon» — это, пожалуй, самая частая просьба специалистов службы техподдержки. Подробный журнал активности ОС удаленного компьютера позволяет устранять неполадки, не мучаясь с брандмауэрами и часовыми поясами. Без этого решать проблемы было бы гораздо труднее. А параметры командной строки (о них речь пойдет ниже) позволяют пользователям, нуждающимся в помощи, получать трассировку, просто запуская командный файл, не выслуживания длительных объяснений процедур сохранения и прочих операций в Procmon.

Сохранение трассировок Procmon

Для сохранения трассировки Procmon нажмите Ctr+S или щелкните значок **Save** на панели инструментов, чтобы открыть диалоговое окно **Save To File (Сохранить в файл)** (см. рис. 4-17).

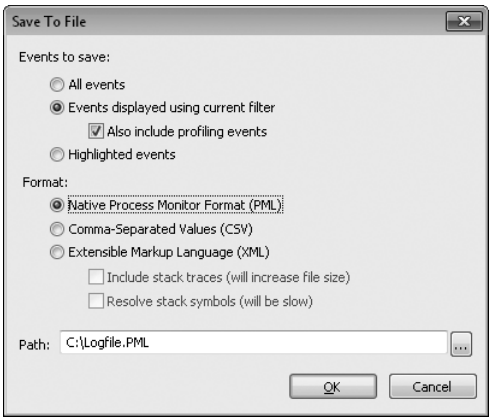


Рис. 4-17. Диалоговое окно Save To File

Вы можете по своему усмотрению сохранять все события (независимо от того, отображаются они или нет), только события, отображаемые текущим фильтром (с событиями трассировки или без них) или же только подсвеченные события.

Procmon может сохранять трассировки в одном из трех форматов. PML является «родным» форматом файла Procmon, который сохраняет все захваченные данные в точности, включая информацию о стеках и модулях, позволяя загружать их в Procmon на любом компьютере. При просмотре в системе с установленными средствами отладки Windows по данными PML-файла можно загружать нужные символы с серверов символов (и двоичные файлы, если трассировка открыта на другом компьютере, подробнее см. в главе 2).

Обратите внимание: внутренний формат PML-файлов различается в x86- и x64-версиях Windows. Формат для x86 читается в обеих версиях, а журналы из 64-разрядных версий Windows — только 64-разрядными версиями (см. раздел о загрузке трассировок).

Другим вариантом является сохранение в файле формата CSV. Такие файлы могут импортироваться в Excel и другие приложения, предназначенные для анализа данных, а также в программы для сравнения текстовых файлов, такие как WinDiff или fc.exe. В CSV-файлах Procmon сохраняет только текстовые данные из колонок, выбранных для отображения. В первой строке CSV-файла содержатся имена полей. Перед сравнением трассировок в формате CSV не забудьте удалить поля вроде **Time Of Day**, которые будут различаться всегда.

Procmon также может сохранять свои данные в формате XML для обработки в программах, поддерживающих разбор XML. Например, следующий сценарий Windows PowerShell разбирает трассировку Procmon в формате XML и выводит упорядоченный список всех уникальных путей модулей, загруженных из любых папок, кроме C:\Windows:

```
$x = [xml]$(gc logfile.xml) $x.SelectNodes("//module")
?{ !$_.Path.ToLower().StartsWith("c:\windows\") }
%{ $_.Path }
sort -Unique
```

Ниже представлен результат обработки этим сценарием 5-Мб XML-файла трассировки:

```
C:\PROGRA~1\WI4EB4~1\wmpband.dll
C:\Program Files\Common Files\microsoft shared\ink\tiptsf.dll
C:\Program Files\Debugging Tools for Windows\DbgHelp.dll
c:\Program Files\Sysinternals\Procmon.exe
C:\Program Files\Virtual Machine Additions\mrxvpcnp.dll
C:\Program Files\Virtual Machine Additions\VMBACKUP.DLL
C:\Program Files\Virtual Machine Additions\vmsrvc.exe
C:\Program Files\Virtual Machine Additions\vmusrvc.exe
C:\Program Files\Virtual Machine Additions\vpcmap.exe
C:\Program Files\Virtual Machine Additions\VPCShExG.dll
c:\program files\windows defender\MpClient.dll
C:\Program Files\Windows Defender\MpRtMon.DLL
c:\program files\windows defender\mprtplug.dll
c:\program files\windows defender\mpsvc.dll
C:\Program Files\Windows Defender\MSASCui.exe
C:\Program Files\Windows Defender\MsMpRes.dll
C:\Program Files\Windows Media Player\wmpnetwk.exe
C:\Program Files\Windows Media Player\WMPNSCFG.exe
C:\Program Files\Windows Media Player\wmpnssci.dll
C:\Program Files\Windows Sidebar\sidebar.exe
C:\ProgramData\Microsoft\Windows Defender\Definition Updates\{02030721-61CF-400A-86EE-
1A0594D4B35E}\mpengine.dll
```

При сохранении в XML возможно включить трассировку стеков с символами. Но имейте в виду, что эти параметры увеличат размер сохраняемого файла и время его сохранения. Не забывайте также, что визуализация больших XML-файлов без схемы в Internet Explorer вводит браузер в «ступор».

Загрузка трассировок Procmon

Procmon, работающий в системе с архитектурой x86, может открывать только трассировки, записанные в системах той же архитектуры. Напротив, Procmon в системе с архитектурой x64 может открывать трассировки как из 32-, так и из 64-разрядных систем. Чтобы открыть трассировку, записанную в x86, в системе x64, Procmon нужно запустить с параметром командной строки **/Run32** — загрузится 32-разрядная версия Procmon. Учтите, что Procmon, запущенный в 32-разрядном режиме в 64-системе, не сможет записывать события. Если Procmon уже работает, откройте диалоговое окно **Open File (Открыть файл)**, щелкнув значок **Open (Открыть)** на панели инструментов. Можно открыть файл журнала с помощью параметра командной строки **/OpenLog**, как показано ниже:

- для загрузки трассировок из x86 в x64:

```
procmon.exe /run32 /openlog logfile.pml
```

- для загрузки на другом компьютере:

```
procmon.exe /openlog logfile.pml
```

При каждом запуске с параметром **/OpenLog** Procmon связывает .PML-файлы с программой Procmon**.** Следовательно, после первого запуска Procmon можно открывать его журналы просто двойным щелчком в **Проводнике**. То же происходит при запуске Procmon с параметром **/Run32**. Параметр **/Run32** будет удален из файловой ассоциации, если запустить Procmon без него.

Procmon не требует прав администратора для загрузки трассировки и не отправляет запросы на повышение прав в Windows Vista и выше при запуске с параметром **/OpenLog**. Однако для записи событий понадобится перезапустить Procmon с правами администратора.

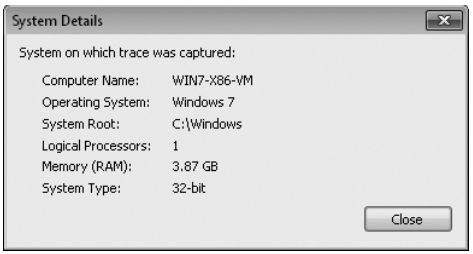


Рис. 4-18. Диалоговое окно System Details

Файл журнала трассировки содержит информацию о системе, в которой были собраны данные, в том числе, имя компьютера, версию операционной

системы, разрядность системы (32 или 64), корневой каталог системы, число процессоров и объем ОЗУ. Все это можно увидеть, открыв окно **System Details (Подробности системы)** (рис. 4-18) меню **Tools (Сервис)**.

Просматривать символы в трассировке стека на компьютере, на котором эта трассировка была записана, можно без установки средств отладки и символов. Напротив, для просмотра такой трассировки на другом компьютере в системе должно быть и то и другое, а также доступ к файлам символов и двоичным файлам на веб-сервере открытых символов Microsoft.

Мониторинг до входа и после выхода из системы

До сего момента предполагалось, что Procmon запускается в интерактивном сеансе, однако Procmon поддерживает и мониторинг активности системы без входа в нее и после выхода пользователей из системы.

Ведение журнала загрузки

Можно настроить Procmon так, чтобы он регистрировал активность системы с самого начала процесса загрузки. Эта функция понадобится для диагностики неполадок, возникающих перед, во время или до входа пользователя. Примером могут быть сбои при загрузке и запуске драйверов устройств, автоматически запускаемых служб, собственно входе или инициализации оболочки. Кроме того, мониторинг загрузки позволяет диагностировать неполадки при выходе пользователя из системы и выключении компьютера.

Ведение журнала загрузки — это единственный режим Procmon, допускающий жесткие перезагрузки, поэтому его можно использовать для диагностики зависаний и крахов системы, в том числе, возникающих при включении и выключении.

Если установить флажок **Enable Boot Logging (Включить журнал загрузки)** в меню **Options**, Procmon настраивает свой драйвер так, что при следующем запуске системы Procmon загружается в самом начале загрузке до большинства других драйверов. Драйвер Procmon регистрирует активность системы в файле %windir%\Procmon.PMB, продолжит регистрировать ее далее и до выключения или до тех пор, пока вы снова не запустите Procmon вручную. Таким образом, если вы не запустите Procmon вручную после загрузки, то сможете записать трассировку всего цикла, от загрузки системы до выключения ПК. В этом случае драйвер Procmon остается загруженным до выключения компьютера.

После загрузки вместе с системой драйвер Procmon снова становится драйвером, загружаемым по требованию. Таким образом, параметр **Enable Boot Logging** остается включенным только на одну загрузку. Активировать мониторинг следующей загрузки вам придется вручную, и так каждый раз.

При запуске Procmon проверяет наличие сгенерированного, но не сохраненного журнала загрузки. Если Procmon найдет его, то спросит, нужно ли сохранить его, и, если да, то где (рис. 4-19). После этого Procmon открыва-

ет и отображает сохраненный журнал. Если вы не сохраните файл журнала трассировки в другом месте, он будет перезаписан при следующем включении мониторинга загрузки.

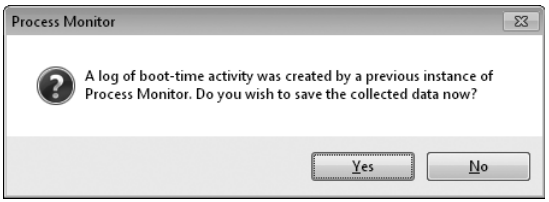


Рис. 4-19. Запрос сохранения трассировки Procmon

Просматривая активность системы во время загрузки, помните, что в начале загрузки работает только процесс System, а его активность фильтруется по умолчанию. Для просмотра активности процесса System выберите **Advanced Output** в меню **Filter**. Обратите внимание, что мониторинг событий сети зависит от параметра **Event Tracing for Windows (ETW)** и недоступен в журналах загрузки. Кроме того, события трассировки процессов и потоков не записываются в журнал загрузки. Наконец, Procmon не включает трассировку загрузки в безопасном режиме.

Если включение трассировки загрузки вызывает сбой, отключите ее, выбрав параметр **Последняя успешная загрузка (Last Known Good)** в меню вариантов загрузки Windows. Для выбора данного параметра нажмите F8 во время запуска Windows.

Продолжение работы Procmon после выхода из системы

Включение трассировки загрузки — единственная возможность вести мониторинг с помощью Procmon и при завершении работы. Если требуется записать события во время или после выхода пользователя из системы, но полной трассировки до выключения ПК не требуется, также используйте трассировку загрузки. Тем не менее, помимо требуемых событий после выхода, в журнале окажутся все данные с момента запуска системы, т.е. вы получите трассировку с большим избытком информации. Другой вариант — запустить Procmon так, чтобы он продолжал работать и после выхода пользователя из системы.

Одним из способов мониторинга выхода пользователей из системы является использование служб терминалов с помощью функций **Быстрого переключения пользователей (Fast User Switching)** или **Удаленного рабочего стола (Remote Desktop)**. Когда пользователь уже вошел в систему, начните новый сеанс под другой записью пользователя и запустите Procmon. Вернитесь в сеанс первого пользователя и выйдите из системы. Вернитесь ко второму сеансу и прекратите запись событий. Установите фильтр на атрибут **Session (Сеанс),** отображающий только события, которые произошли в сеансе первого пользователя.

Другой эффективный способ мониторинга активности после выхода из системы — использование PsExec с параметром -s для запуска Procmon

под системной учетной записью, как неинтерактивной системной службы. Для этого нужно применить кое-какие хитрости, поскольку вы не сможете управлять данным экземпляром Procmon обычным способом:

- необходимо указать файл для записи данных в командной строке с помощью параметра */BackingFile*. Помните, что изменить его до остановки программы вы не сможете, а при перезапуске Procmon под системной записью без указания другого файла поддержки ранее записанная трассировка будет затерта;
- в командной строке должны быть параметры **/AcceptEula** и **/Quiet**, чтобы Procmon не выводил диалоговые окна, которые не удастся закрыть;
- Procmon должен необходимо корректно останавливать. Чтобы сделать это, не выключая систему, нужно выполнить **Procmon /Terminate** тем же образом, что и первую команду.

Более подробно об этих параметрах рассказывается ниже, о сеансах см. в главе 2, а о PsExec — в главе 6.

Ниже приведен пример команды для запуска трассировки Procmon, продолжающейся и после выхода из системы:

```
PsExec -s -d Procmon.exe /AcceptEula /Quiet /BackingFile C:\Procmon.pml
```

Завершает трассировку следующая команда:

```
PsExec -s -d Procmon.exe /AcceptEula /Terminate
```

Параметр *PsExec -d* позволяет завершить PsExec, не дожидаясь завершения конечного процесса.

Экземпляр Procmon, запущенный с помощью PsExec под системной учетной записью, во время завершения работы системы заканчивает регистрацию при уничтожении CSRSS процессов пользовательского режима, продолжить мониторинг дальше позволяет только журнал загрузки.

Длительная трассировка и управление размером журналов

Файлы трассировки Procmon могут быть очень большими, особенно трассировки загрузки и прочих длительных процессов. В таких ситуациях будут полезны функции управления размерами журнала.

Удаление отфильтрованных событий

Обычно Procmon регистрирует всю активность системы, включая события, скрытые фильтрами. Это дает возможность изучать разные аспекты работы системы, изменяя фильтры. Тем не менее, если заранее известно, что часть данных длинной трассировки не потребуется, можно освободить от них журнал с помощью параметра **Drop Filtered Events (Удаление отфильтрованных событий)** в меню **Filter**.

Когда активен параметр **Drop Filtered Events** события, не соответствующие критериям фильтров, не попадают в журнал, что позволяет держать под контролем его размер. Разумеется, эти данные восстановить не удастся. Данный параметр действует только на регистрацию текущих событий, ранее записанные события из журнала не удаляются.

Обратите внимание на то, что фильтрация не действует при трассировке загрузки, поэтому параметр **Drop Filtered Events** не уменьшает размер журнала трассировки загрузки. Однако, фильтры и параметр **Drop Filtered Events** применяются во время последующей обработки журнала загрузки. Поэтому для удаления лишних событий при просмотре активности процесса System и других низкоуровневых событий не забудьте активировать параметр **Enable Advanced Output** в меню **Filter** перед перезагрузкой.

Длина трассировки

Process Monitor отслеживает объем переданной ему памяти и прекращает захват событий, когда виртуальная память системы заканчивается. Открыв диалоговое окно **History Depth (Длина трассировки)** (рис. 4-20) в меню **Options,** можно ограничить число хранящихся записей и оставить Process Monitor работать на длительное время, гарантируя захват последних событий. Число регистрируемых событий варьируется от 1 до 199 миллионов (по умолчанию — 199 миллионов).

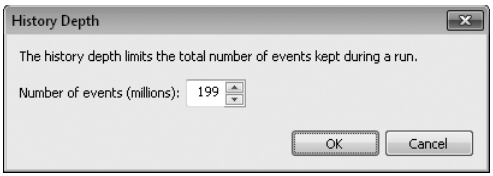


Рис. 4-20. Диалоговое окно History Depth

Файлы для записи данных

По умолчанию Procmon сохраняет зарегистрированные данные в виртуальной памяти. Если виртуальная память заканчивается, Procmon автоматически завершает запись с сообщением об ошибке. Если нужно записать больше данных, чем вмещает виртуальная память, можно настроить Procmon для сохранения данных в файлах на диске. Размер файла, используемого для этой цели, ограничен лишь размером свободного места на жестком диске.

Можно настраивать и просматривать информацию о файлах для записи, выбрав Backing Files **(Резервные файлы)** в меню **File** — откроется диалоговое окно **Process Monitor Backing Files**, изображенное на рис. 4-21. Изменения настройки файлов для записи вступят в силу при записи в новый журнал или после очистки текущего.

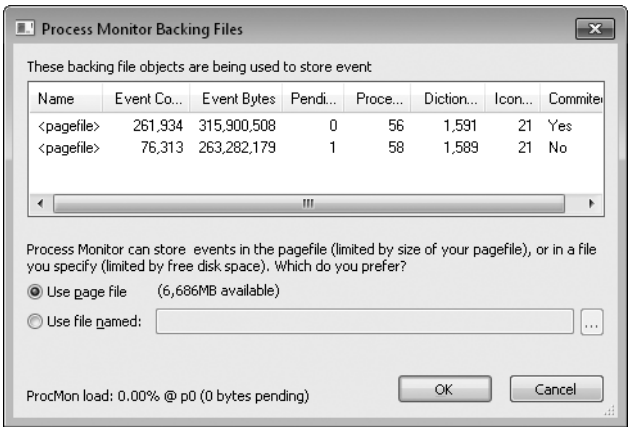


Рис. 4-21. Диалоговое окно Process Monitor Backing Files

Имейте в виду, что при выборе записи в файл Procmon будет создавать дополнительные файлы, чтобы файлы с данными не стали слишком большими. Имена таких файлов генерируются на основе базового имени, к которому прибавляются возрастающие номера (рис. 4-22). Файлы с одинаковым базовым именем, хранящиеся в одной папке, Procmon считает одним журналом.

Диалоговое окно **Process Monitor Backing Files** отображает также информацию по диагностике, в том числе, захваченные события и число наблюдаемых процессов.

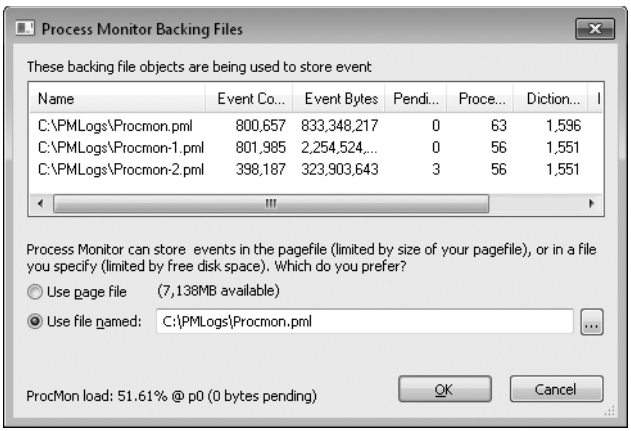


Рис. 4-22. Диалоговое окно Process Monitor Backing Files

Импорт и экспорт параметров конфигурации

Меню **File** позволяет экспортировать в .PMC-файл полную конфигурацию программы, включая фильтры, правила подсветки, отображаемые поля, их порядок и размер, файлы для записи данных, символы, расширенный вывод и сброс отфильтрованных событий. Экспортированную конфигурацию мож-

но импортировать в другую систему или загружать с параметром командной строки **/LoadConfig** (см. следующий раздел). Можно создать несколько ярлыков Procmon с различными файлами конфигурации **/LoadConfig** для различных задач. Фильтры также можно импортировать и экспортировать по отдельности (подробнее см. выше).

Примечание Procmon хранит все свои параметры конфигурации в разделе реестра HKEY CURRENT USER\Software\Sysinternals\Process Monitor. Простейший способ восстановить конфигурацию Procmon по умолчанию — закрыть все экземпляры Procmon, удалить этот раздел реестра и снова запустить Procmon. При просмотре журналов Procmon из систем x86 в 64-разрядных версиях Windows 32-разрядная версия Procmon сохраняет свои параметры конфигурации в HKCU\Software\S$_y$sinternals\ProcessMonitor32.

Автоматизация Procmon: параметры командной строки

Procmon поддерживает параметры командной строки, удобные при вызове программы из сценариев. Допустим, начинающему пользователю нужно запустить Procmon с определенной конфигурацией, чтобы подготовить для вас трассировку. Чтоб не «разжевывать» пользователю настройку и запуск Procmon, можно просто дать ему командный файл с нужными параметрами.

Меню **Help (Справка)** Procmon включает краткий обзор параметров командной строки Procmon. Более подробно они описаны в табл. 4-4.

Табл. 4-4. Параметры командной строки

Параметр	Описание
/OpenLog pm/-f//e	Открывает сохраненный ранее файл журнала (трассировку) Procmon. Обратите внимание: файл журнала должен экземпляром Procmon, совместимым с архитектурой ЦП системы, в которой он был записан
/BackingFile *pml-file*	Сохраняет события в указанном файле. Размер файла ограничен размером свободного места на диске. Имейте в виду, что данный параметр во время работы программы изменить нельзя (подробнее см. выше)
/PagingFile	Сохраняет события в виртуальной памяти, ограниченной системным файлом подкачки. Используется для восстановления для отмены параметра /BackingFile
/NoConnect	Запускает Procmon, но не начинает автоматический захват данных (по умолчанию регистрация событий начинается при запуске программы)
/NoFilter	Сбрасывает фильтр при запуске. Все правила фильтра, кроме исключающих события трассировки, удаляются
/AcceptEula	Подавляет вывод окна с лицензионным соглашением (подразумевает согласие с условиями лицензионного соглашения)
/LoadConfig *config-file*	Загружает сохраненный ранее файл конфигурации (подробнее см. ниже)

Табл. 4-4. (*окончание*)

Параметр	Описание
/Profiling	Активирует функцию Thread Profiling
/Minimized	Запускает Procmon в свернутом виде
/WaitForldle	Ожидает готовности другого экземпляра Procmon на том же рабочем столе Win32 для приема команд. Пример использования данного параметра см. ниже
/Terminate	Прекращает работу всех экземпляров Procmon, работающих на одном рабочем столе Win32, а затем выполняет выход. Этот параметр использует оконные сообщения для отправки команды конечному экземпляру Procmon (см. Раздел Главы 2 «Сеансы, оконные станции, рабочие столы и оконные сообщения»)
/Quiet	Не подтверждает установки фильтров во время запуска. По умолчанию, если правила фильтров были настроены, Procmon отображает диалоговое окно Filter позволяя изменять их перед записью данных
/Run32	Запускает 32-разрядную версию для загрузки 32-разрядных файлов журналов (только в x64)
/HookRegistry	Только в 32-разрядной версии Windows Vista и выше позволяет Procmon использовать перехват системных вызовов вместо обратного вызова для мониторинга активности реестра, что дает возможность видеть виртуальные операции реестра Microsoft Application Virtualization (App-V, ранее Softgrid). Используется при первом запуске Process Monitor в сеансе и только для устранения неполадок в сериализованных приложениях App-V
/SaveAspaf/,	При использовании параметра /OpenLog экспортирует записанный журнал в файл формата XML, CSV, или PML (определяется расширением заданного файла,.xml,.csv или .pml)
/SaveAsl part	При использовании параметра /OpenLog экспортирует трассировку в XML вместе со стеком (см. выше)
/SaveAs2 part	При использовании параметра /OpenLog экспортирует трассировку в XML вместе со стеком и символами (см. выше)

Ниже приведены примеры использования данных параметров:

- Открытие 32-разрядного файла журнала в 64-разрядной версии Windows:

```
Procmon.exe /Run32 /OpenLog c:\pmlLogs\logfile.pml
```

- Вот более сложный пример сценария, регистрирующего операции записи программой Notepad.exe в файле C:\notepad.pml:

```
set PMExe=»C:\Program Files\Sysinternals\Procmon.exe» set PMHide= /AcceptEula /Quiet /
Minimized set PMCfg= /LoadConfig C:\TEMP\PmCfg.pmc set PMFile= /BackingFile C:\notepad.pml
start «» %PMExe% %PMHide% %PMCfg% %PMFile%
%PMExe% /WaitForIdle
```

```
notepad.exe
%PMExe% /Terminate
start «» %PMExe% /PagingFile /NoConnect /Minimized /Quiet
%PMExe% /WaitForIdle
%PMExe% /Terminate
```

Разберем последний пример подробнее.

- Строка 1 *(set PMExe)* заносит в переменную путь к Procmon, чтобы не повторять его ввод в следующих командах.
- Строка 2 *(set PMHide)* задает параметры командной строки, чтобы Procmon не создавал помех пользователю.
- Строка 3 *(set PMCfg)* задает ранее сохраненный файл конфигурации с фильтром, отбирающим события записи для Notepad.exe и удаляющим остальные события.
- Строка 4 *(set PMFile)* настраивает файл для записи.
- Строка 5 использует команду **start** для запуска экземпляра Procmon с немедленным возвратом управления командному файлу.
- Строка 6 запускает второй экземпляр Procmon, который ожидает загрузки первого экземпляра и начала захвата им данных *(параметр /WaitForIdle)*, а затем возвращает управление командному файлу. Затем командой из строки 7 запускается Блокнот. Когда пользователь закроет его, управление вернется командному файлу.
- Строка 8 завершает работу экземпляра Procmon, регистрировавшего события.
- Для восстановления использования файла подкачки в качестве хранилища данных в строке 9 запускается экземпляр Procmon с соответствующим параметром *(/PagingFile)*, который не регистрирует события.
- Когда этот экземпляр приготовится принимать команды (строка 10), его можно завершить (строка 11).

Инструменты для анализа

Procmon поддерживает несколько способов визуализации и простейшего анализа событий, собранных в трассировке. Их можно найти в меню **Tools**:

- **Process Activity Summary (Сводка активности процесса)**
- **File Summary (Сводка по файлам)**
- **Registry Summary (Сводка по реестру)**
- **Stack Summary (Сводка по стеку)**
- **Network Summary (Сводка по сети)**
- **Cross Reference Summary (Сводка по перекрестным ссылкам)**
- **Count Occurences (Счетчик событий)**

Диалоговые окна **Summary** — не модальные, поэтому можно открывать несколько таких окон одновременно, продолжая при этом работать с главным окном.

Сводка активности процесса

Диалоговое окно **Process Activity Summary** (рис. 4-23) отображает таблицу с процессами, для которого были захвачены данные с применением текущего фильтра. Каждая строка таблицы содержит имя и идентификатор процесса, график использования процессора, число файлов, события реестра и сети, пиковые значения транзакций и размера рабочего набора, а также графики, показывающие изменения этих и других параметров во временной шкале процесса. Можно сохранить всю текстовую информацию из этого окна в CSV-файле, щелкнув кнопку **Save**.

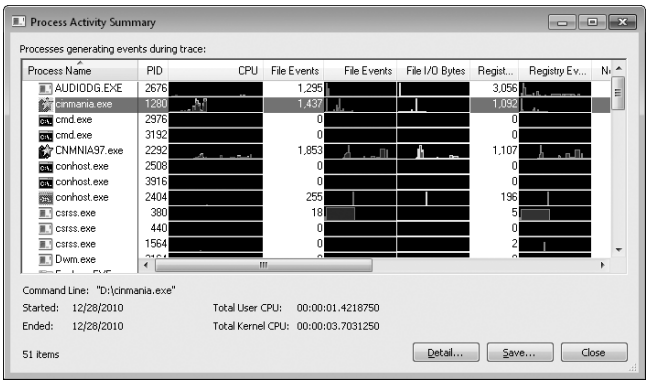


Рис. 4-23. Диалоговое окно Process Activity Summary

При выборе строки отображается дополнительная информация о процессе в нижней части диалогового окна: командная строка, время начала и завершения, а также совокупность времени пользователя и времени ядра ЦП. Если дважды щелкнуть строку или выбрать ее и щелкнуть кнопку **Details**, то появится диалоговое окно **Process Timeline (Шкала времени процесса)** для данного процесса (рис. 4-24). Можно изменять размер колонок или передвигать их, перетаскивая соответствующие части заголовков.

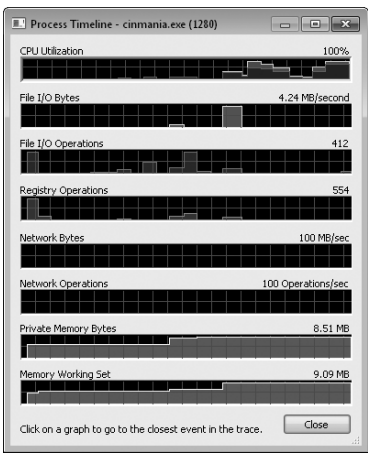


Рис. 4-24. Диалоговое окно Process Timeline

Временная шкала процесса (рис. 4-24) отображает графики процесса из диалогового окна **Process Activity Summary**, расположенные друг над другом в окне с настраиваемым размером. Если щелкнуть точку на графике, то будет выбрано ближайшее соответствующее событие для данного процесса в главном окне. Предположим, например, что на диаграмме появились внезапные всплески активности файловых операций ввода-вывода, пики выделения закрытой памяти и роста рабочего набора. Щелкните пиковую точку на любой диаграмме, и в главном окне Procmon будет выбрано событие, ближайшее к пику для данного процесса.

Сводка по файлам

Диалоговое окно **File Summary**, изображенное на рис. 4-25, систематизирует информацию о каждой операции с файлами и папками, отображаемой текущим фильтром, и группирует результаты на отдельных вкладках по пути, папке и расширению файла. Для каждого уникального пути файловой системы диалоговое окно показывает полное затраченное время при выполнении файлового ввода-вывода, а также сколько раз файл был открыт, закрыт, прочитан и записан, получен и установлен ACL файла и пр. Кроме того, отображается общее число выполненных операций, а также число байтов, прочитанных и записанных в файл.

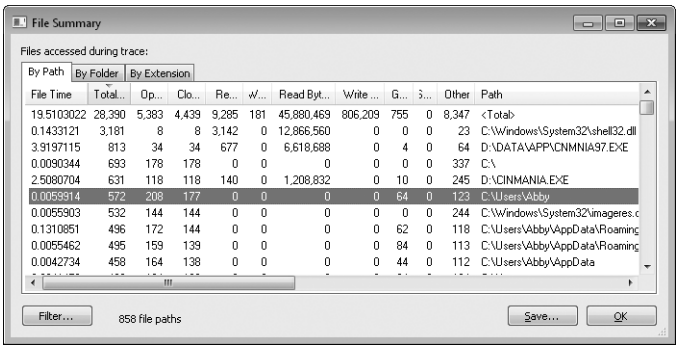


Рис. 4-25. Вкладка By Path диалогового окна File Summary

Вкладка **By Path (По пути)** отображает простой список, в котором каждый уникальный путь появляется в отдельной строке.

Вкладка **By Folder (По папке)** (рис. 4-26) отображает папки в виде дерева. Узлы, которые можно раскрывать, содержат сводку по операциям, выполненным над объектами в этой папке. Узлы, которые раскрывать нельзя, просто показывают данные операций, выполненных над некоторым объектом. Например, в узле папки могут быть два узла исполняемых файлов. Узел, который можно раскрыть, показывает операции, выполненные над папкой в целом, а раскрываемый узел содержит сводки операций, выполненных над отдельными вложенными в нее папками и файлами.

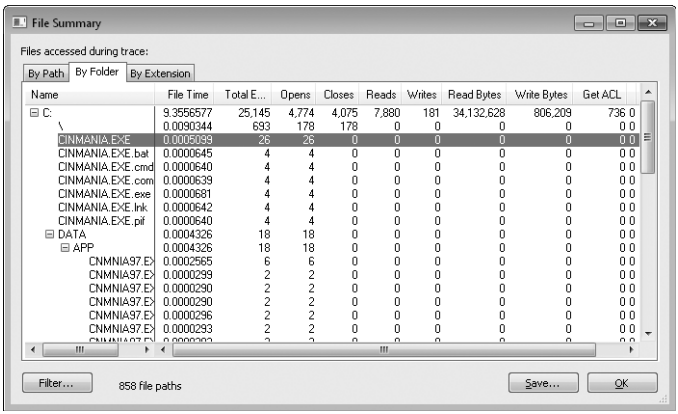


Рис. 4-26. Вкладка By Folder диалогового окна File Summary

Вкладка **By Extension (По расширению)** (см. рис. 4-27) отображает одноуровневое дерево для каждого расширения файла: если развернуть узел для расширения, то появится список всех файлов с данным расширением в виде дочерних узлов. Строка с именем расширения содержит сводку данных для всех файлов с данным расширением.

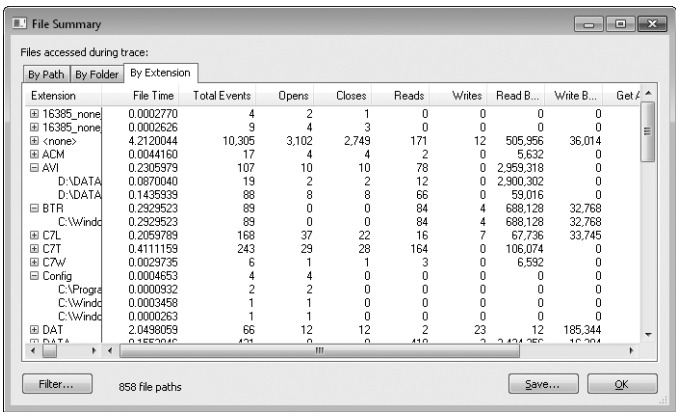


Рис. 4-27. Вкладка By Extension диалогового окна File Summary

Если щелкнуть заголовок таблицы, то таблица в данной вкладке будет отсортирована по соответствующему полю. Содержимое вкладок **By Folder** и **By Extension** сортируется соответствующим образом. Сортировка позволят быстро выявлять закономерности в работе системы. Так, отсортированные строки на вкладке **By Folder** позволяют определить, с какими папками и файлами ввод-вывод (по числу операций, числу прочитанных или записанных байтов либо по времени ввода-вывода) выполняется наиболее интенсивно. Сортировка колонок на вкладке **By Extension** показывает, к какому типу файлов чаще всего обращаются. Можно также изменить расположение столбцов списка, перетащив их заголовки (на вкладках **By Folder** и **By Extension** слева столбцы фиксированы).

Двойной щелчок строки в списке добавляет к текущему фильтру правило **Path** на основе пути, зарегистрированного в данной строке. Если щелкнуть кнопку **Filter**, откроется диалоговое окно **Filter**, где можно продолжить настройку фильтра.

Кнопка **Save** на каждой из вкладок сохраняет таблицу в текущем виде в формате CSV.

Сводка по реестру

Подобно окну **File Summary**, диалоговое окно **Registry Summary** (рис. 4-28) отображает все пути реестра, на которые ссылаются операции с реестром в таблице, а также общее время операций ввода-вывода с разделом, число открытых, закрытых, прочитанных, записанных и прочих элементов, а также сумму по всем операциям. Если щелкнуть заголовок столбца, таблица будет отсортирована по его содержимому, порядок столбцов также можно изменить путем перетаскивания их заголовков. Двойной щелчок строки добавляет правило **Path** для пути регистра к текущему фильтру в данной строке. Окно **Filter** вызывается щелчком кнопки **Filter**, данные можно сохранить в CSV-файле.

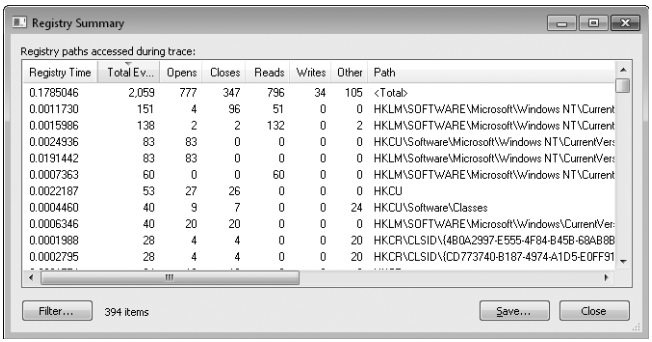


Рис. 4-28. Диалоговое окно Registry Summary

Сводка по стеку

Диалоговое окно **Stack Summary** (рис. 4-29) записывает трассировки стека для событий Procmon, выявляет их сходства и различия и визуализирует результаты в виде деревьев. Для каждого фрейма в стеке вызовов можно увидеть, сколько событий трассировки он сгенерировал, узнать общее время, затраченное на регистрацию, имя и путь модуля, а также абсолютное смещение в нем. При наличии символов (подробнее см. в главе 2) **Stack Summary** также показывает имена функций, путь к файлам исходного текста и номера строк в них для каждого фрейма стека.

Примечание **Stack Summary** не охватывает весь код и не регистрирует полный профиль исполняемой программы. Выводится только число трассируемых событий Procmon, время ЦП, затраченное на выполнении этих операций, и проценты от суммарных показателей.

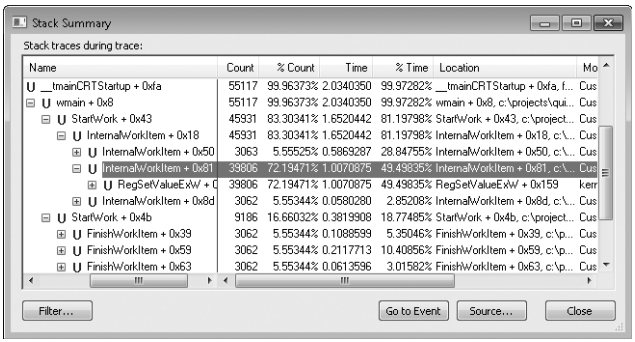


Рис. 4-29. Диалоговое окно Stack Summary

На рис. 4-29 показана сводка по стеку для программы, для которой в системе есть полный набор символов. Два верхних фрейма показывают, что библиотечная функция CRT *tmainCRTStartup* вызвала стандартную входную функцию *wmain*, а вызванные ей функции привели к возникновению 55117 событий, записанных Procmon с текущим фильтром. Раскрывая наиболее часто встречающиеся узлы, легко определить, какой код исполняется наиболее интенсивно. Свыше 72% событий, отображенных с текущим фильтром, сгенерировано вызовами *InternalWorkItem+0x81* и *RegSetValueExW* (39806 событий).

Если выбрать фрейм стека и щелкнуть кнопку **Go To Event**, то будет выбрано первое событие в трассировке с соответствующим стеком вызовов. Кнопка **Source** активируется при наличии полного набора символов для выбранного элемента. Если доступен файл исходного текста, то щелчок кнопки **Source** выводит соответствующие его строки в окне просмотра.

Как и в других сводных окнах, поля упорядочиваются щелчками заголовков, но крайнее левое поле можно только перетаскивать.

Имейте в виду, что генерации сводки по стеку требует времени, особенно если выполняется интерпретация символов.

Сводка по сети

Диалоговое окно **Network Summary** (рис. 4-30) отображает все конечные точки и порты TCP- и UDP-подключений в отфильтрованной трассировке, а также число установленных и разорванных соединений, отправленных и полученных сообщений, общее число вышеуказанных событий, а также отправленных и полученных байтов. По щелчку заголовка столбца таблица сортируется по его содержимому, также возможно перетаскивание столбцов на новое место. Двойной щелчок строки добавляет к текущему фильтру правило **Path** для данной конечной точки и порта. Окно **Filter** вызывается щелком кнопки **Filter** и позволяет сохранять содержимое в CSV-файл.

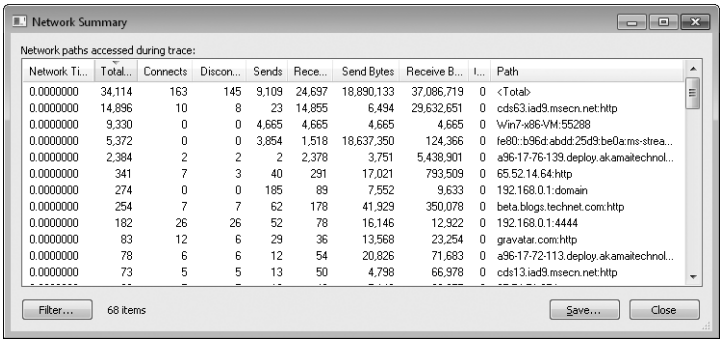


Рис. 4-30. Диалоговое окно Network Summary

Сводка по перекрестным ссылкам

Диалоговое окно **Cross Reference Summary** (рис. 4-31) содержит все отображаемые текущим фильтром пути к объектам, к которым обращаются несколько процессов. Каждая строка содержит путь, а также перечень читающих и записывающих процессов. Допускается перетаскивание столбцов и сохранение данных CSV-файл. Двойной щелчок строки или выделение строки с щелчком кнопки **Filter On Row (Фильтр по строке)** добавляет выбранный путь к фильтру.

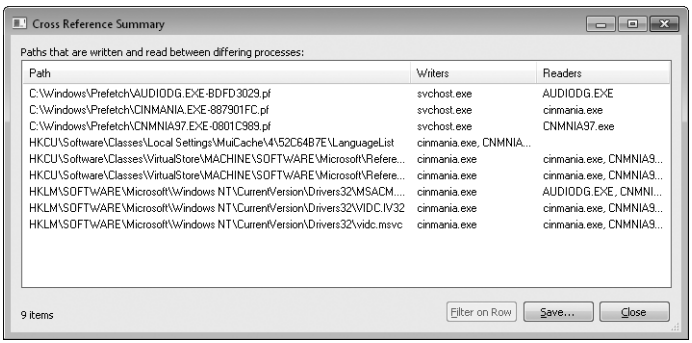


Рис. 4-31. Диалоговое окно Cross Reference Summary

Счетчик событий

Выберите имя столбца в диалоговом окне **Count Values Occurrences (Значения счетчика событий)** (рис. 4-32), и щелкните кнопку **Count (Подсчитать).** Procmon отображает все значения выбранного атрибута, а также число событий, в которое данное значение входит при использовании текущего фильтра. Допускается перетаскивание столбцов и сохранение данных CSV-файл. Двойной щелчок объекта добавляет в фильтр правило по данному полю и значению.

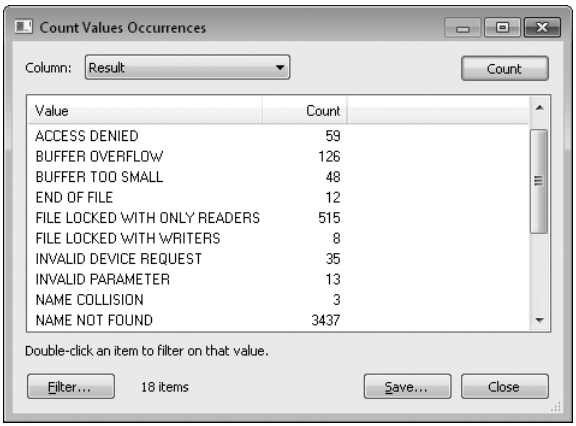


Рис. 4-32. Диалоговое окно Count Values Occurrences

Включение вывода отладчика в трассировки Procmon

Procmon поддерживает API, позволяющим разработчикам создавать события вывода отладчика, добавляемые в поток событий с пользовательским комментарием. Например, можно включать пользовательский вывод отладчика в трассировку при вызове и завершении функций, чтобы сопоставить эти события с активностью файловой системы, реестра и другими событиями. С помощью фильтров **Exclude Events Before** и **Exclude Events After** для выбранных событий отладки можете легко выделить интересующие вас аспекты программы. В отличие от стандартного вывода отладчика Windows, который записывается утилитой DebugView (см. главу 7), и других отладчиков, данный API-интерфейс предназначен специально для работы с Procmon.

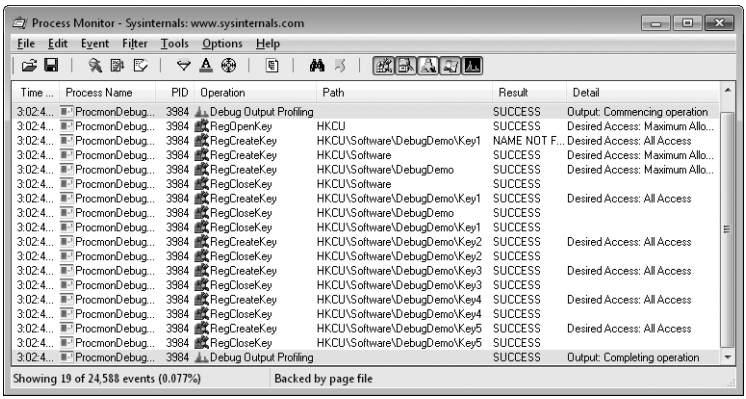


Рис. 4-33. События вывода отладчика

Эти события генерируются отладчиком и относятся к классу событий трассировки, наряду с событиями трассировки процесса и потока. Имейте

в виду, что по умолчанию все события трассировки блокируются фильтром. Для просмотра событий вывода отладчика установите переключатель **Show Profiling Events (Показать события трассировки)** на панели инструментов. После этого можно включить подсветку операции вывода отладчика и скрыть события трассировки процесса. На рис. 4-33 отображен подсвеченный вывод отладчика, чередующийся с обращениями к реестру.

Все процессы, в том числе, работающие с низким IL, могут использовать данную API-функцию, принимающую строки Unicode длиной до 2048 знаков. Вот пример:

```
#include <stdio.h> #include <windows.h>
const ULONG FILE_DEVICE_PROCMON_LOG = 0x00009535; const ULONG IOCTL_EXTERNAL_LOG_
DEBUGOUT =
(ULONG) CTL_CODE( FILE_DEVICE_PROCMON_LOG, 0x81, METHOD_BUFFERED, FILE_WRITE_ACCESS );
BOOL WriteProcmonDebugOutput(const wchar_t * szDebugOutput) {
if (!szDebugOutput)
return FALSE; HANDLE hDevice = CreateFileW( L»\\\\.\\Global\\ProcmonDebugLogger»,
GENERIC_READ | GENERIC_WRITE,
FILE_SHARE_READ | FILE_SHARE_WRITE | FILE_SHARE_DELETE,
NULL,
OPEN_EXISTING,
FILE_ATTRIBUTE_NORMAL,
NULL ); if ( hDevice == INVALID_HANDLE_VALUE )
return FALSE; DWORD buflen = wcslen(szDebugOutput) * sizeof(wchar_t); DWORD unused =
0; BOOL ret = DeviceIoControl( hDevice, IOCTL_EXTERNAL_LOG_DEBUGOUT,
(LPVOID)szDebugOutput, buflen, NULL, 0, &unused, NULL ); CloseHandle(hDevice); return
ret; }
```

Гуру отладки Джон Роббинс разработал для нее оболочку, которую можно без труда встроить в свои «родные» или управляемые приложения. Ее можно скачать по следующей ссылке: *http://www.wintellect.com/downloads/ProcMonDebugOutput.zip*.

Панель инструментов

В этом разделе разъясняется назначение кнопок на панели инструментов Procmon и также даются ссылки на темы с описанием соответствующих функций.

Рис. 4-34. Панель инструментов Procmon

Значки панели инструментов Procmon, изображенной на рис. 4-34, слева направо:

- **Open Log** — загрузка сохраненных трассировок Procmon»;
- **Save Log** — сохранение трассировок Procmon;
- **Capture Events** — начало работы с Procmon;

- **Autoscroll** — начало работы с Procmon;
- **Clear Display** — начало работы с Procmon;
- **Диалоговое окно Filter** — фильтрация и подсветка;
- **Highlight dialog box** — фильтрация и подсветка;
- **Include Process From Window** — фильтрация и подсветка;
- **Show Process Tree** — дерево процессов;
- **Find** — поиск событий;
- **Jump To Object** — переход к разделу реестра или файлу;
- **Show/Hide Registry Activity (переключатель)** — фильтрация и подсветка;
- **Show/Hide File System Activity (переключатель)** — фильтрация и подсветка;
- **Show/Hide Network Activity (переключатель)** — фильтрация и подсветка;
- **Show/Hide Process And Thread Activity (переключатель)** — фильтрация и подсветка;
- **Show/Hide Profiling Events (переключатель)** — отображение событий трассировки.

Глава 5

Autoruns

Я часто слышу такой вопрос: «Что это за *штуки* работают у меня на компьютере?» А за ним часто идет следующий: «Как убрать все это?» Windows — платформа с большими возможностями в плане расширения. Программисты могут писать не только приложения, которые пользователи запускают вручную, но и «более удобные» программы, которые запускаются автоматически, не утруждая пользователя выбором, нужно ли запускать их или нет. Достигают этого, используя видимые и невидимые функции Проводника, Internet Explorer и драйверов, управляющих оборудованием. Впрочем, нередко целесообразность автозапуска с точки зрения пользователя в лучшем случае сомнительна. Иногда это делается для удобства кого-то совершенно постороннего, при этом автоматически запускаемая программа действует во вред пользователю (такую программу называют *вредоносной*; см. главу 18).

Программы, запускающиеся автоматически, т.е. без вмешательства пользователя, ниже относятся к категории *автозапуска*. Сюда входят драйверы и службы, запускаемые при загрузке компьютера; приложения, утилиты и расширения оболочки, запускающиеся при входе пользователя в систему; расширения обозревателя, загружающиеся при запуске Internet Explorer, и не только. Вообще, в 32-разрядных версиях Windows более сотни мест в файловой системе и реестре, обеспечивающих автозапуск программ, а в x64-версиях их еще больше. Эти места часто называют *точками расширения для автозапуска* (Autostart Extensibility Points, ASEP).

Точки ASEP создавались с разумными и полезными целями. Например, если вы хотите, чтобы люди, с которыми вы общаетесь с помощью мгновенных сообщений, знали, когда вы в сети, удобен автозапуск IM-клиента при входе в систему. Многим пользователям нравятся поисковые панели и средства для просмотра PDF, ставшие частью Internet Explorer. Да и немалая часть самой Windows реализована в виде драйверов, служб и расширений Проводника, загружаемых через ASEP.

С другой стороны, тем же способом запускаются многочисленные пробные программы, которыми поставщики начиняют свои новые компьютеры, забивая область уведомлений панели задач. Так же создаются «полускрытые» процессы, которые помогают разработчикам программ быстрее запу-

скать их приложения. Но нужны ли вам все эти процессы, отъедающие системные ресурсы? Кстати, почти любое вредоносное ПО использует точки ASEP, и почти все ASEP Windows используются той или иной вредоносной программой.

В Windows есть утилита **Настройка системы (System Configuration)** (msconfig.exe, рис. 5-1), позволяющая отследить некоторые точки автозапуска, но она показывает лишь небольшую их часть, а ее возможности ограничены. Кроме того, msconfig требует прав администратора даже для просмотра параметров. Значит, с помощью этой утилиты нельзя ни увидеть, ни отключить автозапуск программ, настроенный под записями *пользователей*, не входящих в группу Администраторы.

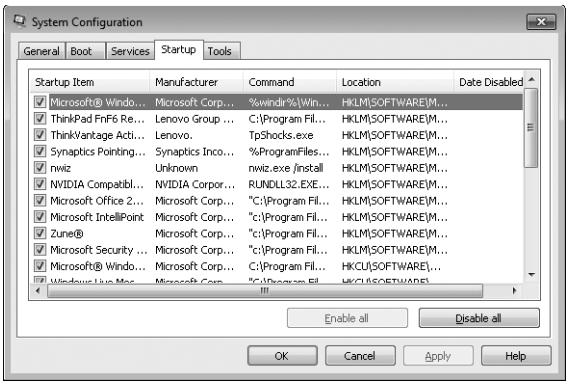


Рис. 5-1. Утилита MSConfig из Windows показывает очень ограниченный набор параметров автозапуска

Мы с Брюсом создали утилиту Autoruns, которая показывает столько точек автозапуска, сколько мы нашли, и позволяет отключать или удалять их. Сведения, предоставляемые этой утилитой, можно собрать и вручную, если вы знаете, в каких местах реестра и файловой системы их искать. Наша утилита автоматизирует эту задачу, проверяя большое число ASEP за считанные секунды и выявляя подозрительные точки, например, не имеющие цифровой подписи. Также мы разработали версию Autoruns для командной строки, AutorunsC. Она позволяет делать то же самое с помощью сценариев.

С помощью Autoruns или AutorunsC легко получить эталонный набор ASEP в системе, чтобы потом сравнивать с ним результаты новых проходов для обнаружения изменений и решения проблем. Многие компании используют Autoruns в своих системах управления изменениями для регистрации новых эталонных параметров после каждого обновлении образа рабочей системы.

Введение в Autoruns

Autoruns сразу после запуска начинает заполнять экран записями известных ASEP. Как показано на рис. 5-2, каждая подсвеченная строка представ-

ляет ASEP со значком Regedit, если эта ASEP находится в реестре, или со значком папки, если ASEP ссылается на каталог файловой системы[1]. Ниже подсвеченной строки идут записи из данной точки ASEP. Каждая строка содержит имя записи автозапуска, описание, название издателя и путь к исполняемому файлу, а также значок этого файла. В каждой строке имеется флажок, позволяющий временно отключить запись. Панель в нижней части окна содержит сведения о выбранной записи, включая полную команду запуска. На вкладке **Everything (Все)**, которая открывается при запуске Autoruns, отображаются все ASEP в системе; остальные 17 вкладок позволяют просматривать отдельные категории точек автозапуска, подробнее о них — ниже в этой главе.

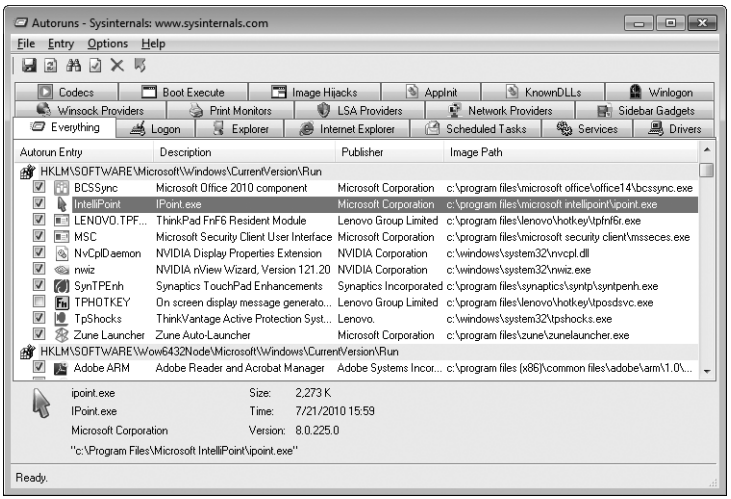


Рис. 5-2. Главное окно Autoruns

В столбце **Image Path (Путь к образу)** указан полный путь к автоматически запускаемому файлу. В некоторых случаях имя этого файла идет первым в командной строке автозапуска. Если при автозапуске используется хост-процесс (такой как Cmd.exe, Wscript.exe, Rundll32.exe или Svchost.exe), в пути к образу будет указан запускаемый сценарий или DLL, а не файл хост-программы. Если запускаемый файл в указанном месте не обнаружен, в столбце **Image Path** будет текст **File not found (Файл не найден)**.

Для файлов, содержащих сведения о версиях, таких как EXE и DLL, столбцы **Description (Описание)** и **Publisher (Издатель)** заполняются данными из полей **Description (Описание)** и **Company Name (Название компании)**. Если цифровая подпись файла проверена, столбец **Publisher (Издатель)** будет отображать имя владельца соответствующего сертификата (см. ниже).

---

[1] Запланированные задачи (Scheduled Tasks) отображаются со значком папки, так как параметры конфигурации задач до Windows Vista хранились в папке %windir%\Tasks. В новой ОС Планировщик хранит эти сведения в разделе реестра HKLM\SOFTWARE\Microsoft\ Windows NT\ CurrentVersion\Schedule\TaskCache.

Столбцы **Description** и **Publisher** остаются пустыми, если целевой файл не найден, если в сведениях о версиях нет полей **Description** и **Company Name**, и если сведений о версиях нет вообще (в случае сценариев так почти всегда и бывает).

Для того чтобы быстро найти точку, нажмите Ctrl+F и введите искомый текст — Autoruns выделит найденную строку, содержащую этот текст. Чтобы найти следующий подходящий элемент, нажмите F3. Чтобы скопировать выделенную строку в буфер обмена как текст, разделенный знаками табуляции, нажмите Ctrl+C.

Пока Autoruns сканирует систему, меню **Options (Параметры)** недоступно. Если требуется изменить параметры (см. ниже), отмените сканирование, нажав клавишу Esc. Изменение любых параметров в этом меню (кроме шрифта) вступит в силу при следующем сканировании. Изменив параметры, нажмите F5 или щелкните кнопку **Refresh (Обновить)** на панели инструментов, чтоб начать новое сканирование.

Отключение и удаление элементов автозапуска

Autoruns позволяет отключать и удалять элементы автозапуска. Элементы удаляются безвозвратно, так что удаляйте их, только если хотите навсегда отключить автозапуск приложения. Выберите элемент в списке и нажмите клавишу Del. Отменить удаление невозможно, поэтому перед удалением элемента появится запрос подтверждения.

Если отключить элемент, сняв флажок рядом с ним, Autoruns сохранит данные для восстановления этого элемента. Например, для ASEP из реестра Autoruns создает в исходном разделе подраздел AutorunsDisabled, в который копирует отключенный параметр, прежде чем удалить его. Windows не обрабатывает параметры этого подраздела, так что элементы из него запускаться не будут, но Autoruns будет отображать их как отключенные элементы автозапуска. Если снова установить флажок, элемент вернется на исходное место. Для точек ASEP, расположенных в файловой системе, таких как меню **Пуск (Start)**, Autoruns создает скрытую папку **AutorunsDisabled**, в которую перемещает удаленные элементы.

Обратите внимание на то, что удаление или отключение элемента автозапуска предотвращает автоматический запуск этого элемента в будущем, но не завершает уже запущенные процессы.

Также учтите, что отключение элементов автозапуска, важных для загрузки, инициализации и восстановления системы может так испортить систему, что восстановить ее будет почти невозможно.

Autoruns и администраторские разрешения

Большинство точек ASEP находится в местах, к которым обычным пользователям разрешен доступ только для чтения. В некоторых версиях Windows разделы реестра, содержащие сведения о конфигурации некоторых служб,

заблокированы, и многие из запланированных задач недоступны для чтения пользователями. Однако Autoruns обычно прекрасно работает «на просмотр» и без администраторских прав. Права администратора требуются для просмотра полного списка элементов автозапуска и для изменения состояния системных элементов, таких как куст HKLM и папка **Автозагрузка (Startup)** для всех пользователей. При попытке удалить, отключить или включить такой элемент без прав администратора вы получите сообщение об отказе в доступе.

В Windows Vista и выше окно с сообщением об ошибке содержит кнопку **Run as Administrator (Запуск от имени администратора)**, позволяющую перезапустить Autoruns с повышенными правами (рис. 5-3). С администраторскими правами изменение конфигурации должно пройти успешно. Можно также перезапустить Autoruns с повышенными правами при помощи механизма UAC, выбрав в меню **File (Файл)** команду **Run as Administrator**.

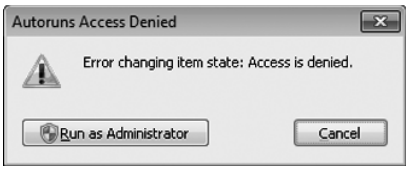


Рис. 5-3. Отказ в доступе и возможность запустить Autoruns с повышенными правами

Чтобы гарантированно запускать Autoruns с повышенными правами, запускайте утилиту из командной строки с параметром `-e`. В Windows Vista и последующих версиях этот параметр запрашивает повышение привилегий UAC, если они еще не повышены. В Windows XP и Windows Server 2003 появится диалоговое окно **Запуск от имени (Run As)**. Установите переключатель в положение **учетную запись другого пользователя (The following user)** и введите учетные данные пользователя, имеющего права администратора. Подробнее о запуске от имени других пользователей и повышении прав через UAC см. в главе 2.

Проверка подписей кода

В принципе любой, написав программу, может вставить в нее название «Microsoft Corporation». И если в столбце **Publisher (Издатель)** указано именно это, вовсе не факт, что этот файл скомпилировали в Microsoft и с тех пор не изменяли. Определенную уверенность в подлинности источника и целостности файла дает проверка связанной с ним цифровой подписи. Некоторые файловые форматы позволяют внедрять в файл цифровую подпись. Также подпись, необходимая для проверки содержимого одного файла, может находиться в другом файле — так можно подписывать даже простейшие текстовые файлы.

Чтобы проверить цифровую подпись элемента, выберите его и нажмите Ctrl+V. Если файл подписан с действительным сертификатом от корневого

центра сертификации, считающегося надежным на текущем компьютере, в столбце **Publisher** появится текст **(Verified) [(Проверено)]**, за которым будет идти имя владельца сертификата, которым подписан код. Если файл не был подписан или проверка не по каким-то причинам не прошла, появится текст **(Not verified) [(Не проверено)]**, за которым будет идти название компании, взятое из сведений о версиях файла (если оно там указано).

Чтобы не проверять элементы по одному, можно включить функцию проверки подписей кода (**Verify Code Signatures**) в меню **Options (Параметры)** и повторить сканирование, нажав F5, — Autoruns попытается проверить подписи для всех элементов автозапуска. Сканирование может продлиться дольше, так как при этом проверяется отзыв всех сертификатов, а для этого требуется хорошая связь с Интернетом.

Файлы, подписи которых не прошли проверку, можно считать подозрительными, так как вредоносное ПО часто внедряют, устанавливая файлы, имитирующих системные файлы Windows.

Утилита Sysinternals Sigcheck, описанная в главе 8, предоставляет более подробные сведения о подписях файлов, включая тип подписи и ее расположение.

Как скрыть системные ASEP Microsoft

По умолчанию выводится длинный список ASEP, так как сама Windows, как мы уже говорили, активно использует этот механизм. Обычно системные элементы автозапуска не предоставляют интереса при решении проблем. И в элементах автозапуска, созданных другими программами от Microsoft, такими как приложения пакета Microsoft Office, также можно не искать. Эти элементы автозапуска можно скрыть, включив параметр **Hide Windows Entries (Скрывать элементы Windows)** и **Hide Microsoft And Windows Entries (Скрывать элементы Microsoft и Windows)** в меню **Options**, обновив затем окно нажатием F5. Параметр **Hide Windows Entries** включен по умолчанию.

Работа этих параметров зависит также от того, включен ли параметр **Verify Code Signatures**. Если проверка подписей не включена, параметр **Hide Windows Entries** скрывает все элементы, в имени целевого файла которых присутствует слово Microsoft, а также элементы, целевой файл которых находится в папке %windir%. Параметр **Hide Microsoft And Windows Entries** проверяет только название «Microsoft» в поле **Company Name** и пропускает такие элементы. Как уже говорилось, легко создать программу, проходящую такую проверку, поэтому рекомендуется включать параметр **Verify Code Signatures**.

Если проверка подписей включена, параметр **Hide Windows Entries** скрывает элементы, подписанные сертификатом кода Windows (компоненты Windows подписаны особым сертификатом, отличным от подписи других продуктов Microsoft.) Параметр **Hide Microsoft And Windows Entries**

скрывает элементы, подписанные любым сертификатом Microsoft, выданным центром сертификации, считающимся надежным в данной системе.

Примечание Некоторые файлы, поставляемые с Windows, в частности, драйверы, созданы сторонними производителями, и в поле **Company Name** у них указано название соответствующей компании, но они дополнительно подписываются с использованием сертификата Windows. Следовательно, эти элементы могут быть скрыты, когда включена проверка подписей, но отображаться, когда проверка не проводится.

Параметры, отвечающие за проверку и маскировку системных ASEP, сохраняются в реестре и продолжают действовать, когда тот же пользователь снова запускает Autoruns.

Получение дополнительной информации об ASEP

Если щелкнуть элемент правой кнопкой, откроется меню **Entry (Элемент)**. Команды из этого меню используют другие программы для отображения дополнительной информации о выбранном в окне Autoruns элементе:

- **Properties (Свойства)** — открывает одноименное окно Проводника для файла, на который указывает путь элемента ASEP;
- **Jump To (Перейти к)** — открывает папку или раздел реестра, в котором находится элемент автозапуска. В случае реестра открывается Regedit.exe и выполняется эмуляция нажатия клавиш для перехода к целевой записи (если нужный раздел в Regedit не открылся, повторите команду **Jump To**). Для точек ASEP из файловой системы эта команда открывает в Проводнике соответствующую папку. Для запланированных задач открывается Планировщик задач (Task Scheduler). Обратите внимание: в Windows Vista и выше для перехода по разделам в Regedit уровень целостности (IL) Autoruns должен быть не ниже, чем у Regedit;
- **Search Online (Поиск онлайн)** — выполняет поиск в Интернете по имени файла, используя обозреватель и поисковую систему по умолчанию;
- **Process Explorer** — если путь образа ведет к исполняемому файлу (а не к сценарию или файлу DLL), и процесс с таким именем еще работает, Autoruns пытается вызвать Procexp, чтобы тот открыл окно свойств для этого процесса. Чтобы эта команда работала, Procexp должен находиться в одной папке с Autoruns или быть доступным по одному из стандартных путей, либо должен быть уже запущен. При этом уровень прав у Procexp не должен быть выше, чем у Autoruns, в противном случае эта команда не будет работать.

Просмотр элементов автозапуска для других пользователей

Если Autoruns работает с правами администратора, появляется меню **User (Пользователь)**, содержащее список имен пользователей, входивших в систему, чьи профили доступны. Если выбрать имя пользователя в этом меню, система будет просканирована заново для поиска точек ASEP этого поль-

зователя, включая разделы Run ветви HKCU реестра этого пользователя и папку **Автозагрузка (Startup)** из его профиля.

Такая возможность полезна, например, для поиска вредоносного ПО, установленного обычными пользователями. Под учетной записью обычного пользователя можно изменять только пользовательские ASEP. Приложения, имеющие только стандартные пользовательские привилегии, не могут изменять общесистемные параметры или параметры учетных записей системы и других пользователей. Чтобы не входить в систему, дав тем самым возможность запуска этому вредоносному ПО (и, возможно, помешав сканированию Autoruns), можно войти в систему под учетной записью администратора. Далее нужно запустить Autoruns, выбрать скомпрометированную учетную запись в меню **User (Пользователь)**, проверить точки ASEP этого пользователя и удалить вредоносные элементы.

Просмотр точек ASEP отключенной системы

Autoruns позволяет просматривать точки ASEP отключенного экземпляра Windows из-под другой, стабильно работающей системы Windows. Это может быть полезно:

- даже если эта система с Windows не работает, в ней можно обнаружить и удалить сбойные или неправильно настроенные точки ASEP;
- вредоносные программы, в частности, руткиты, могут мешать Autoruns точно определять точки ASEP. Например, руткит, перехватывающий и изменяющий обращения к реестру, может скрывать содержимое некоторых разделов от Autoruns. Если завершить работу ОС и изучить ее точки ASEP в другой, не зараженной системе, эти элементы не будут скрыты;
- вредоносные файлы могут имитировать подпись надежного издателя, на самом же деле корневой сертификат получен от злоумышленника. В стабильной системе, где нет поддельных сертификатов, проверка фальшивых подписей не пройдет.

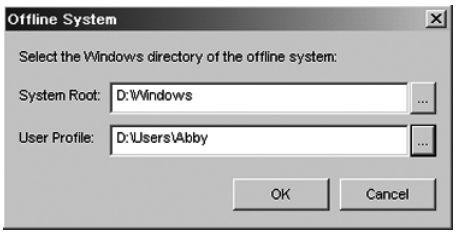


Рис. 5-4. Выбор системных папок и профиля в отключенной системе

Чтобы выполнить анализ отключенной системы, необходимо запустить Autoruns с административными правами и предоставить доступ к файловой системе отключенной копии Windows. Выберите в меню **File (Файл)** команду **Analyze Offline System (Анализировать отключенную систему)** и укажите системную папку целевой системы Windows и папку профиля поль-

зователя (рис. 5-4). Заметьте, что носитель с файлами реестра должен быть доступен для записи.

Перечисление неиспользуемых точек ASEP

По умолчанию, Autoruns отображает строки только для тех точек ASEP, в которых имеются элементы автозапуска. Если в меню **Options** включить параметр **Empty Locations (Пустые места)**, будут показаны все проверяемые точки ASEP независимо от наличия в них элементов автозапуска. Autoruns проверяет множество точек ASEP, так что выходной список значительно увеличится. Этот параметр применяют, чтобы убедиться, что проверяются нужные точки ASEP, или просто из любопытства. После включения параметр **Empty Locations** применяется при следующем сканировании.

Изменение шрифта

Чтобы изменить шрифт, используемый Autoruns для отображения результатов, выберите в меню **Options** команду **Font (Шрифт)**. Изменение шрифта вступает в силу незамедлительно.

Категории автозапуска

При первом запуске Autoruns все элементы автозапуска отображаются в одном длинном списке на вкладке **Everything**. Окно программы содержит до 17 других вкладок с разными категориями элементов из полного списка (рис. 5-5).

Рис. 5-5. Категории автозапуска отображаются на вкладках

Категория Logon

На этой вкладке перечислены «стандартные» элементы автозапуска, обрабатываемые при запуске Windows и входе пользователя в систему. Точки ASEP, указанные на этой вкладке, используются приложениями чаще всего. Сюда входят различные разделы реестра Run и RunOnce, папки **Автозагрузка (Startup)** из меню **Пуск (Start)** и сценарии, выполняющиеся при загрузке, выключении, входе и выходе из системы. Здесь также указаны исходные процессы сеансов пользователей, такие как Userinit и оболочка рабочего стола. Эти точки ASEP включают как пользовательские, так и общесистемные места, а также элементы, управление которыми выполняется через групповую политику. Здесь же перечислены разделы установщика и установленных компонентов, которые никогда не вносились документацию и не раскрывались сторонним производителям, но были обнаружены ими и стали применяться как во благо, так и во вред.

Ниже перечислены точки ASEP из категории **Logon**, которые Autoruns проверяет в 64-разрядных версиях Windows 7.

Папка Автозагрузка (Startup) в меню Пуск (Start) всех пользователей
%ALLUSERSPROFILE%\Microsoft\Windows\StartMenu\Programs\Startup
Папка Автозагрузка (Startup) в меню Пуск (Start) пользователя
%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup
Пользовательские ASEP из раздела HKCU\Software
HKCU\Software\Microsoft\Windows\CurrentVersion\Run
HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce
HKCU\Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Microsoft \Windows\CurrentVersion\Run
HKCU\Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\ Microsoft\Windows\CurrentVersion\Runonce
HKCU\Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\ Microsoft\Windows\CurrentVersion\RunonceEx
HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\Load
HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\Run
HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell
Пользовательские ASEP из раздела HKCU\Software, предназначенные для управления через групповую политику
HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System\Shell
HKCU\Software\Policies\Microsoft\Windows\System\Scripts\Logon
HKCU\Software\Policies\Microsoft\Windows\System\Scripts\Logoff
Общесистемные ASEP из реестра
HKLM\Software\Microsoft\Windows\CurrentVersion\Run
HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce
HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnceEx
HKLM\SOFTWARE\Microsoft\Active Setup\Installed Components
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\ Microsoft\Windows\CurrentVersion\Run
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\ Microsoft\Windows\CurrentVersion\Runonce
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\ Microsoft\Windows\CurrentVersion\RunonceEx
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\AppSetup
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Taskman

HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit
HKLM\SYSTEM\CurrentControlSeAControl\SafeBoot\AIternateShell
HKLM\System\CurrentControlSet\Control\Terminal Server\Wds\rdpwd\StartupPrograms
Общесистемные ASEP из реестра, предназначенные для управления через групповую политику
HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System\Shell
HKLM\Software\Policies\Microsoft\Windows\System\Scripts\Logon
HKLM\Software\Policies\Microsoft\Windows\System\Scripts\Logoff
HKLM\Software\Policies\Microsoft\Windows\System\Scripts\Startup
HKLM\Software\Policies\Microsoft\Windows\System\Scripts\Shutdown
HKLM\Software\Microsoft\Windows\CurrentVersion\Group Policy\Scripts\Startup
HKLM\Software\Microsoft\Windows\CurrentVersion\Group Policy\Scripts\Shutdown
Общесистемные ASEP из реестра (только 64-разрядные системы)
HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Run
HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\RunOnce
HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\RunOnceEx
HKLM\SOFTWARE\Wow6432Node\Microsoft\ActiveSetup\lnstalled Components
Общесистемные ASEP ActiveSync из реестра
HKLM\SOFTWARE\Microsoft\Windows CE Services\AutoStartOnConnect
HKLM\SOFTWARE\Microsoft\WindowsCEServices\AutoStartOnDisconnect
Общесистемные ASEP ActiveSync из реестра, только 64-разрядные системы
HKLM\SOFTWARE\Wow6432Node\Microsoft\WindowsCEServices\AutoStartOnConnect
HKLM\SOFTWARE\Wow6432Node\Microsoft\WindowsCEServices\AutoStartOnDisconnect

Категория Explorer

На этой вкладке перечислены общие элементы автозапуска, связанные напрямую с Проводником Windows, обычно запускаемые в процессе Explorer.exe. Большинство этих элементов — общесистемные, но среди них есть и несколько пользовательских. Ключевые элементы с этой вкладки:

- расширения оболочки, добавляющие элементы в контекстные меню, изменяющие страницы свойств и управляющие отображением столбцов в окнах папок;
- расширения пространств имен, такие как **Рабочий стол (Desktop)**, **Панель управления (Control Panel)** и **Корзина (Recycle Bin)**, а также расширения пространств имен сторонних производителей;

- подключаемые обработчики пространств имен, обрабатывающие стандартные протоколы, такие как http, ftp и mailto, а также расширения Microsoft или сторонних производителей, такие как about, mk и res;
- подключаемые фильтры MIME.

В 64-разрядных версиях Windows внутрипроцессные компоненты, такие как библиотеки DLL, могут загружаться только в процессы, созданные для той же архитектуры процессора. Например, расширения оболочки, реализованные как 32-разрядные DLL, могут загружаться только в 32-разрядную версию Проводника, а в 64-разрядных системах по умолчанию используется 64-разрядная версия Проводника. Поэтому не все расширения работают в 64-разрядных системах.

Ниже приведен список расположений ASEP из категории **Explorer**, которые Autoruns проверяет в 64-разрядных версиях Windows 7.

Пользовательские ASEP из раздела HKCU\Software
HKCU\Software\Classes*\ShellEx\ContextMenuHandlers
HKCU\Software\Classes*\ShellEx\PropertySheetHandlers
HKCU\Software\Classes\AIIFileSystemObjects\ShellEx\ContextMenuHandlers
HKCU\Software\Classes\AIIFileSystemObjects\ShellEx\DragDropHandlers
HKCU\Software\Classes\AIIFileSystemObjects\ShellEx\PropertySheetHandlers
HKCU\Software\Classes\Directory\Background\ShellEx\ContextMenuHandlers
HKCU\Software\Classes\Directory\ShellEx\ContextMenuHandlers
HKCU\Software\Classes\Directory\Shellex\CopyHookHandlers
HKCU\Software\Classes\Directory\Shellex\DragDropHandlers
HKCU\Software\Classes\Directory\Shellex\PropertySheetHandlers
HKCU\Software\Classes\Folder\Shellex\ColumnHandlers
HKCU\Software\Classes\Folder\ShellEx\ContextMenu Handlers
HKCU\Software\Classes\Folder\ShellEx\DragDropHandlers
HKCU\Software\Classes\Folder\ShellEx\ExtShellFolderViews
HKCU\Software\Classes\Folder\ShellEx\PropertySheetHandlers
HKCU\SOFTWARE\Classes\Protocols\Filter
HKCU\SOFTWARE\Classes\Protocols\Handler
HKCU\Software\Microsoft\Ctf\LangBarAddin
HKCUXSOFTWAREXMicrosoftMnternetExplorerXDesktopXComponents
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ShelllconOverlay ldentifiers
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\ShellServiceObject DelayLoad

Общесистемные ASEP из реестра
HKLM\Software\Classes*\ShellEx\ContextMenuHandlers
HKLM\Software\Classes*\ShellEx\PropertySheetHandlers
HKLM\Software\Classes\AllFileSystemObjects\ShellEx\ContextMenuHandlers
HKLM\Software\Classes\AllFileSystemObjects\ShellEx\DragDropHandlers
HKLM\Software\Classes\AllFileSystemObjects\ShellEx\PropertySheetHandlers
HKLM\Software\Classes\Directory\Background\ShellEx\ContextMenuHandlers
HKLM\Software\Classes\Directory\ShellEx\ContextMenu Handlers
HKLM\Software\Classes\Directory\Shellex\CopyHookHandlers
HKLM\Software\Classes\Directory\Shellex\DragDropHandlers
HKLM\Software\Classes\Directory\Shellex\PropertySheetHandlers
HKLM\Software\Classes\Folder\Shellex\ColumnHandlers
HKLM\Software\Classes\Folder\ShellEx\ContextMenuHandlers
HKLM\Software\Classes\Folder\ShellEx\DragDropHandlers
HKLM\Software\Classes\Folder\ShellEx\ExtShellFolderViews
HKLM\Software\Classes\Folder\ShellEx\PropertySheetHandlers
HKLM\SOFTWARE\Classes\Protocols\Filter
HKLM\SOFTWARE\Classes\Protocols\Handler
HKLM\Software\Microsoft\Ctf\LangBarAddin
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\SharedTaskScheduler
HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\ShellExecuteHooks
HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Shelllcon Overlayldentifiers
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\ShellServiceObjectDelayLoad
Общесистемные ASEP из реестра (только в 64-разрядных системах)
HKLM\Software\Wow6432Node\Classes\Directory\Shellex\CopyHookHandlers
HKLM\Software\Wow6432Node\Classes\Directory\Shellex\DragDropHandlers
HKLM\Software\Wow6432Node\Classes\Directory\Shellex\PropertySheetHandlers
HKLM\Software\Wow6432Node\Classes\Folder\Shellex\ColumnHandlers
HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Explorer\SharedTaskScheduler
HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Explorer\ShellExecuteHooks
HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Explorer\ShelllconOverlayldentifiers
HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\ShellServiceObjectDelayLoad

Категория Internet Explorer

Обозреватель Internet Explorer предоставляет немалые возможности для расширения: интерфейсы для работы с панелями, такими как **Избранное (Favorites)** и **История (History)**, панелями инструментов, настраиваемыми элементами меню и кнопками. А вспомогательные объекты обозревателя (Browser Helper Objects, BHO) предоставляют почти неограниченные возможности для расширения функциональности Internet Explorer.

Поскольку пользователи проводят много времени за работой в обозревателе, и через него проходит много конфиденциальной информации (такой как пароли и данные кредитных карт), он стал основной целью злоумышленников. Те же программные интерфейсы, что обеспечивают интеграцию со сторонними программами для чтения документов и обмена сообщениями, используются и для кражи информации, рассылки спама и других вредоносных действий.

Ниже приведен список расположений ASEP из категории **Internet Explorer**, которые Autoruns проверяет в 64-разрядных версиях Windows 7.

Пользовательские ASEP из раздела HKCU\Software
HKCU\Software\Microsoft\Internet Explorer\Explorer Bars
HKCU\Software\Microsoft\Internet Explorer\Extensions
HKCU\Software\Microsoft\InternetExplorer\UrlSearchHooks
Общесистемные ASEP из реестра
HKLM\Software\Microsoft\Internet Explorer\Explorer Bars
HKLM\Software\Microsoft\Internet Explorer\Extensions
HKLM\Software\Microsoft\InternetExplorer\Toolbar
HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects
Пользовательские и общесистемные ASEP из реестра (только в 64-разрядных системах)
HKCU\Software\Wow6432Node\Microsoft\Internet Explorer\Explorer Bars
HKCU\Software\Wow6432Node\Microsoft\Internet Explorer\Extensions
HKLM\Software\Wow6432Node\Microsoft\Internet Explorer\Explorer Bars
HKLM\Software\Wow6432Node\Microsoft\Internet Explorer\Extensions
HKLM\Software\Wow6432Node\Microsoft\Internet Explorer\Toolbar
HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects

Категория Scheduled Tasks

На вкладке **Scheduled Tasks (Запланированные задания)** отображаются элементы, запускаемые Планировщиком Windows (Task Scheduler). Планировщик заданий запускать программы по фиксированному графи-

ку или при возникновении некоторых событий, таких как вход пользователя в систему или простой компьютера в течение определенного времени. Команды, выполняемые At.exe, также входят в этот список. В Windows Vista Планировщик значительно усовершенствован, и Windows активно использует его, так что список элементов на этой вкладке будет довольно длинным, если не скрыть системные элементы Windows.

Так как задачи можно отключать (в отличие от элементов меню Пуск), при снятии флажка в Autoruns задача Планировщика отключается, а не копируется в резервную папку[2].

Если в меню элемента запланированного задания выбрать команду, Autoruns откроет Планировщик, но переход к выбранному элементу выполнен не будет.

Категория Services

Службы Windows работают в неинтерактивных процессах в режиме пользователя, которые можно настроить для запуска независимо от входа пользователей в систему. Управление службами выполняется при помощи стандартного интерфейса Диспетчера служб (Service Control Manager). Один процесс может совместно использоваться несколькими службами. Наглядный пример — Svchost.exe (хост-процесс для служб Windows), предназначенный для служб, реализованных в отдельных библиотеках DLL.

Службы настраиваются в подразделах HKLM\System\CurrentControlSet\Services. Значение **Start** в каждом подразделе определяет, будет ли запускаться служба, и если будет, то как.

На вкладке **Services (Службы)** перечислены службы, которые не были отключены, если только они не отключены с помощью Autoruns (это можно понять по присутствию значения AutorunsDisabled в том же разделе). Содержимое поля **Description (Описание)** берется из текста или ресурса, указанного значением **Description** в разделе конфигурации. В столбце пути к образу показан путь к исполняемому файлу службы; для службы, работающей в Scvhost, — путь к DLL, указанный значением **ServiceDll** из раздела реестра для этой службы или его подраздела **Parameters**. Для просмотра раздела **Parameters** некоторых служб в некоторых версиях Windows требуются администраторские права, если их нет, в таких случаях Autoruns показывает путь к Svchost.exe.

Отключайте или удаляйте автозапуск служб только со знанием дела. Неправильные действия могут снизить производительность системы, сделать ее нестабильной или привести к невозможности загрузки. Опять же, заметьте, что отключение или удаление элемента службы не останавливает эту службу, если она уже запущена.

---

[2] Во время написания книги задания At нельзя было отключать в Windows Vista и выше ни с помощью Autoruns, ни через Планировщик, но можно было удалять.

Один из приемов, используемых вредоносным ПО, — запуск служб, имитирующих системные службы Windows, но таковыми не являющихся, например файл *svchost.exe* из папки Windows, а не System32. Другой прием — ставить настоящие системные службы в зависимость от поддельной службы. Удаление или отключение такой службы без исправления зависимостей может привести к невозможности загрузки системы. Команда **Jump To** в Autoruns очень удобна для проверки зависимостей служб в реестре перед внесением изменений.

Категория Drivers

Драйверы также настраиваются в подразделах HKLM\System\CurrentControlSet\Services, но запускаются в режиме ядра, становясь частью ядра операционной системы. Большинство драйверов установлено в папке **System32\Drivers** и имеет расширение **.sys**. Драйверы позволяют Windows взаимодействовать с различными типами оборудования, включая мониторы, запоминающие устройства, устройства для чтения смарт-карт и т.д. Также они используются антивирусными программами (и утилитами Sysinternals, такими как Procmon и Procexp!) для мониторинга сетевого трафика и файлового ввода-вывода. И, конечно, они используются вредоносным ПО, в особенности, руткитами.

Как и в случае со службами, на вкладке **Drivers (Драйверы)** показаны драйверы, не помеченные как отключенные, если только они не были отключены средствами Autoruns. Значение **Description** берется из сведений о версиях файла драйвера, а путь к образу указывает расположение этого файла.

Большинство сбоев с «синим экраном» вызвано некорректными операциями, выполняемыми в режиме ядра, что в большинстве случаев вызвано ошибками в драйверах сторонних производителей. Гораздо реже причиной появления «синего экрана» является сбойное оборудование, завершение критически важного процесса, такого как Csrss.exe, или преднамеренное действие, вызванное специальной функцией драйвера клавиатуры (см. статью 244139 из базы знаний Knowledge Base по адресу *http://support.microsoft.com/kb/244139*).

При помощи Autoruns можно отключить или удалить проблемный драйвер. Обычно для этого требуется перезагрузка. И в этом случае отключать и удалять конфигурацию драйверов можно только со знанием дела. Многие драйверы критически важны для операционной системы, и любая ошибка в конфигурации может привести к тому, что Windows перестанет работать.

Категория Codecs

В категории **Codecs (Кодеки)** перечислен исполняемый код, загружаемый приложениями для воспроизведения звука и видео. Сбойные и неправильно настроенные кодеки могут приводить к замедлению работы системы и другим проблемам, а их точки ASEP также могут использоваться вредоносным ПО. Ниже перечислены разделы, отображающиеся на этой вкладке.

Разделы из HKLM и HKCU
\Software\Classes\CLSID\{083863Fl-70DE-lld0-BD40-00A0C911CE86}\lnstance
\Software\Classes\CLSID\{7ED96837-96F0-4812-B211-F13C24117ED3}\lnstance
\Software\Classes\CLSID\{ABE3B9A4-257D-4B97-BDlA-294AF496222E}\lnstance
\Software\Classes\CLSID\{AC757296-3522-4Ell-9862-C17BE5A1767E}\lnstance
\Software\Classes\Filter
\Software\Microsoft\Windows NT\CurrentVersion\Drivers32
Разделы из HKLM и HKCU в 64-разрядных версиях Windows
\Software\Wow6432Node\Classes\CLSID\{083863Fl-70DE-lld0-BD40-00A0C911CE86}\lnstance
\Software\Wow6432Node\Classes\CLSID\{7ED96837-96F0-4812-B211-F13C24117ED3}\lnstance
\Software\Wow6432Node\Classes\CLSID\{ABE3B9A4-257D-4B97-BDlA-294AF496222E}\lnstance
\Software\Wow6432Node\Classes\CLSID\{AC757296-3522-4Ell-9862-C17BE5A1767E}\lnstance
\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Drivers32

Категория Boot Execute

На этой вкладке отображаются собственные исполняемые файлы Windows, запускаемые Диспетчером сеансов (Session Manager, Smss.exe) во время загрузки системы. Элемент **BootExecute** обычно содержит задания, которые не могут быть выполнены, когда Windows уже работает, такие как проверка и восстановление жесткого диска (Autochk.exe). Элементы **Execute**, **SOInitialCommand** и **SetupExecute** должны быть пустыми после завершения установки Windows. Ниже перечислены разделы, отображающиеся на этой вкладке.

Разделы на вкладке Boot Execute (Выполнение во время загрузки)
HKLM\System\CurrentControlSet\Control\ServiceControlManagerExtension
HKLM\System\CurrentControlSet\Control\Session Manager\BootExecute
HKLM\System\CurrentControlSet\Control\Session Manager\Execute
HKLM\System\CurrentControlSet\Control\Session Manager\SOInitialCommand
HKLM\System\CurrentControlSet\Control\Session Manager\SetupExecute

Категория Image Hijacks

«Краденными» образами я называю точки ASEP, выполняющие не ту программу, которую указал пользователь, и выполнения которой он ожидает. На вкладке **Image Hijacks («Краденые» образы)** показываются три типа таких перенаправлений:

- **exefile (Файл «exe»)** — изменение ассоциации типа файлов .exe или .cmd с исполняемой командой. Через пользовательский интерфейс Windows

невозможно изменить ассоциации файлов .exe и .cmd, но они могут быть изменены через реестр. Обратите внимание на то, что есть и общесистемные версии этих ASEP;

- **Command Processor\Autorun (Автозапуск процессора команд)** — командная строка, выполняемая при запуске нового экземпляра Cmd.exe. Эта команда выполняется в контексте нового экземпляра Cmd.exe. Существуют пользовательская и общесистемная версии, а также отдельная версия для 32-разрядного варианта Cmd.exe в 64-разрядных системах;
- **Image File Execution Options (IFEO) [Параметры выполнения файла образа]** — содержимое этого раздела используется для некоторых внутренних недокументированных нужд. Одно из *документированных* назначений подразделов IFEO — возможность указать альтернативную программу, запускаемую при запуске определенного приложения. Если создать в этом разделе подраздел с именем файла исходной программы и значением **Debugger**, указывающим путь к альтернативной программе, то вместо исходной будет выполняться альтернативная программа. При этом она получит в качестве параметров командной строки путь и параметры командной строки исходной программы. Этот механизм создавался для того, чтобы запускать программы с отладчиком автоматически. Однако никто не обязывает указывать в качестве альтернативной программы именно отладчик или выполнять строго определенные операции с передаваемыми параметрами командной строки. С помощью этого механизма утилита Process Explorer (см. главу 3) замещает Диспетчер задач.

Ниже перечислены разделы реестра, соответствующие точкам ASEP, показанным на вкладке **Image Hijacks**.

Разделы, проверяемые на предмет «кражи» файлов EXE
HKCU\SOFTWARE\Classes\Exefile\Shell\Open\Command\(Default)
HKCU\Software\Classes\.exe
HKCU\Software\Classes\.cmd
HKLM\SOFTWARE\Classes\Exefile\Shell\Open\Command\(Default)
HKLM\Software\Classes\.exe
HKLM\Software\Classes\.cmd
Разделы автозапуска обработчика команд
HKCU\Software\Microsoft\Command Processor\Autorun
HKLM\Software\Microsoft\Command Processor\Autorun
HKLM\Software\Wow6432Node\Microsoft\Command Processor\Autorun
Разделы, проверяемые на предмет кражи параметров выполнения файла образа
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options
HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Image File Execution Options

Категория AppInit

Библиотеки DLL AppInit создавались разработчиками Windows NT 3.1 с исключительно благородной целью: укажите одну или несколько DLL в разделе реестра Appinit_Dlls, и эти DLL будут загружаться в каждый процесс, загружающий библиотеку user32.dll (т.е. практически в каждый процесс Windows, работающий в режиме пользователя). Ну, и что же здесь не так? А вот что:

- DLL загружаются в процесс во время инициализации User32, т.е. когда выполняется функция *DllMain*. Разработчикам же однозначно сказано, что нельзя загружать другие DLL при помощи *DllMain*. Это может приводить к взаимным блокировкам, неправильному порядку загрузки и, в итоге, к сбоям приложений. Однако AppInit DLL совершает именно этот грех, и действительно, он приводит к блокировкам и сбоям приложений;
- библиотека DLL, автоматически загружаемая в каждый процесс — мечта создателей вредоносных программ. Хотя библиотеки AppInit применяется в «законопослушных» (но неправильно написанных) приложениях, часто ими пользуются вредоносные программы.

Из-за этих проблем библиотеки AppInit DLL не рекомендованы к использованию и по умолчанию отключены в Windows Vista и выше. Для обеспечения обратной совместимости их можно включить, но это не рекомендуется.

Разделы элементов AppInit
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Windows\Appinit_Dlls
HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\ Windows\Appinit_Dlls

Категория KnownDLLs

Функция KnownDLLs помогает повысить производительность системы, гарантируя, что все процессы Windows используют одинаковые версии заданных DLL, а не загружают их из разных каталогов. Во время загрузки Диспетчер сеансов размещает библиотеки DLL, перечисленные в разделе HKLM\System\ CurrentControlSet\Control\Session Manager\KnownDlls, в памяти как именованные объекты-разделы. Когда загружается новый процесс, обращающийся к этим DLL, он использует существующие разделы, а не ищет их снова в файловой системе.

На вкладке **KnownDLLs (Известные DLL)** должны отображаться только проверяемые DLL Windows. В 64-разрядных системах на этой вкладке указана одна точка ASEP, но файлы дублируются для 32- и 64-разрядных версий DLL и размещаются в папках, указанных значениями реестра **DllDirectory** и **DllDirectory32**.

Чтобы убедиться, что вредоносные программы не удалили элемент из этого раздела для загрузки поддельной системной DLL, сохраните результаты проверки Autoruns на подозрительной системы и сравните их с результатами, полученными в надежном экземпляре той же ОС (подробнее см. ниже).

Категория Winlogon

На этой вкладке показаны элементы, связанные с процессом Winlogon.exe, который управляет пользовательским интерфейсом для входа в систему. В Windows XP и Windows 2003 сюда входит интерфейс GINA DLL и пакеты уведомлений Winlogon. В Windows Vista эти интерфейсы удалены, а вместо них предоставлен интерфейс Credential Provider (Поставщик учетных данных).

На этой вкладке показана также выбранная пользователем экранная заставка, запускаемая Winlogon.exe после указанного периода бездействия.

Ниже перечислены разделы реестра, отображающиеся на вкладке **Winlogon**.

Пользовательская экранная заставка
HKCU\Control Panel\Desktop\Scrnsave.exe
Пользовательская экранная заставка, назначаемая через групповую политику
HKCU\Software\Policies\Microsoft\Windows\Control Panel\Desktop\Scrnsave.exe
GINA, пакеты уведомлений и т.д. (Windows XP и Windows Server 2003)
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\SaveDumpStart
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\System
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Taskman
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\UIHost
Точки ASEP интерфейса Credential Provider (Windows Vista и последующие версии)
HKLM\Software\Microsoft\Windows\CurrentVersion\Authentication\Credential Provider Filters
HKLM\Software\Microsoft\Windows\CurrentVersion\Authentication\Credential Providers
HKLM\Software\Microsoft\Windows\CurrentVersion\Authentication\PLAP Providers
Общесистемная идентификация программы для подтверждения успешной загрузки
HKLM\System\CurrentControlSet\Control\BootVerificationProgram\ImagePath

Категория Winsock

Сокеты Windows (Winsock) — расширяемый интерфейс API, позволяющий сторонним производителям добавлять *поставщиков служб передачи данных*, взаимодействующих с Winsock по другим протоколам или уровням поверх существующих протоколов для обеспечения работы приложений-посредников. Сторонние производители могут также добавлять *поставщиков служб пространств имен*, расширяя возможности Winsock по разрешению имен. Поставщики служб подключаются к Winsock посредством *интерфейса поставщиков служб* (Service Provider Interface, SPI). После регистрации в Winsock провайдер служб передачи используется для реализации функций сокетов, таких как *connect* и *accept*, для типов адресов, которые этот провайдер указал как обрабатываемые им. Нет никаких ограничений на то, как поставщик служб передачи реализует свои функции, но реализация обычно включает взаимодействие с драйвером передачи в режиме ядра.

На вкладке **Winsock** перечислены поставщики, зарегистрированные в системе, включая встроенные в Windows. Последние можно скрыть, применив параметры **Hide Windows Entries** и **Verify Code Signatures**, чтобы выделить элементы, которые с большей вероятностью могут вызывать проблемы.

Разделы элементов поставщика Winsock
HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\NameSpace_ Catalog5\ Catalog_Entries
HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\NameSpace_ Catalog5\ Catalog_Entries64
HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\Protocol_ Catalog9\ Catalog_Entries
HKLM\System\CurrentControlSet\Services\WinSock2\Parameters\Protocol_ Catalog9\ Catalog_Entries64

Категория Print Monitors

Элементы, перечисленные на вкладке **Print Monitors (Мониторы печати)**, являются библиотеками DLL, указанными в подразделах HKLM\System\ CurrentControlSet\Control\Print\ Monitors. Эти DLL загружаются в службу Spooler, которая запускается под учетной записью LocalSystem.

Примечание Одна из наиболее распространенных проблем спулера — некорректная реализация мониторов портов сторонними производителями. При решении проблем со спулером печати разумно начать с отключения мониторов портов сторонних производителей, а затем проверить, сохранилась ли проблема.

Категория LSA Providers

Эта категория элементов автозапуска включает пакеты, определяющие или расширяющие средства аутентификации пользователей Windows при помо-

щи подсистемы локального центра безопасности (Local Security Authority, LSA). Если не установлены сторонние пакеты аутентификации, в этот список должны входить только проверяемые элементы Windows. Библиотеки DLL, указанные этими элементами, загружаются процессом Lsass.exe или Winlogon.exe и выполняются под учетной записью LocalSystem.

Точка ASEP **SecurityProviders**, также показанная на этой вкладке, содержит список зарегистрированных поставщиков средств шифрования. Библиотеки DLL из этой точки загружаются во многие процессы привилегированных и обычных пользователей, так что она часто подвергается атакам. (На самом деле, эта ASEP не имеет отношения к LSA, за исключением того, что, как и LSA, предоставляет функции, относящиеся к безопасности.)

Разделы поставщиков средств аутентификации
HKLM\System\CurrentControlSet\Control\Lsa\Authentication Packages
HKLM\System\CurrentControlSet\Control\Lsa\Notification Packages
HKLM\System\CurrentControlSet\Control\Lsa\Security Packages Keys
Разделы зарегистрированных поставщиков средств шифрования
HKLM\System\CurrentControlSet\Control\SecurityProviders\SecurityProviders

Категория Network Providers

На вкладке **Network Providers (Поставщики сети)** перечислены установленные поставщики, управляющие сетевым взаимодействием, указанные в разделе HKLM\System\CurrentControlSet\Control\NetworkProvider\Order. Например, в настольных ОС Windows эта вкладка включает стандартных поставщиков, предоставляющих доступ к серверам SMB (файловым и печати), серверам Microsoft RDP (службы терминалов и удаленный рабочий стол), а также доступ к серверам WebDAV. Дополнительные поставщики могут появляться, если сеть более разнородна и требуется подключаться к другим типам серверов. Все элементы из этого списка должны быть проверяемыми.

Категория Sidebar Gadgets

В Windows Vista и выше эта вкладка содержит список гаджетов боковой панели (в Windows 7 они стали называться гаджетами рабочего стола), появляющихся на рабочем столе пользователя. Хотя гаджеты часто (но не всегда) устанавливаются в системные папки, такие как %ProgramFiles%, конфигурация запускаемых гаджетов находится в файле %LOCALAPPDATA%\Microsoft\Windows Sidebar\Settings.ini, который имеется у каждого пользователя и не передается на другие системы. При отключении и удалении гаджетов при помощи Autoruns изменяются параметры файла Settings.ini. Путь к образу обычно указывает на XML-файл. Гаджеты, входящие в состав Windows, подписаны и могут быть проверены.

Сохранение и сравнение результатов проверки

Результаты работы Autoruns можно сохранить на диск в двух форматах: текст, разделенный знаками табуляции, и двоичный файл, сохраняющий все полученные данные. Двоичные файлы можно загружать в Autoruns для просмотра в будущем и (или) на другом ПК, а также сравнивать с другим набором результатов Autoruns.

В обоих случаях результаты можно только читать; с их помощью нельзя вернуть систему в исходное состояние или восстановить конфигурацию, и в них нельзя добавлять или удалять параметры, изменяя сохраненные результаты.

Сохранение в виде текста с разделителями

Щелкните кнопку **Save (Сохранить)** на панели инструментов. В диалоговом окне сохранения файлов в списке **Тип файла (Save As)** выберите **Text (*.txt) [Текст (*.txt)]** и укажите имя файла, в который хотите сохранить текущие результаты. Данные, показанные на вкладке **Everything (Все)**, будут записаны в файл в виде четырех столбцов, разделенных знаками табуляции. Выходные данные содержат точки ASEP (выделенные строки) в первом столбце и пустые строки в остальных трех. Перед включенными элементами (с установленными флажками) записывается знак плюс (+), перед отключенными — символ «X».

Полученный текстовый файл можно импортировать в Excel. Для первого столбца следует выбрать тип **Текст (Text)**, а не **Общий (General)**, чтобы плюсы в начале строк не интерпретировались как инструкции и спецсимволы.

В этом формате учитываются параметры, выбранные в меню **Options**. Если включен параметр **Include Empty Locations**, в файл будут записаны все точки ASEP, включая те, в которых нет элементов. Если включены параметры **Hide Microsoft and Windows Entries** и **Hide Windows Entries**, соответствующие элементы не будут записаны в файл. Если включен параметр **Verify Code Signatures**, столбец **Publisher** будет включать значения **Verified** либо **Not Verified**.

Autoruns не может загружать результаты, сохраненные в текстовом виде.

Далее в этой главе, в разделе, посвященном AutorunsC, показан способ сохранения результатов в других текстовых форматах при помощи сценариев.

Сохранение в двоичном формате (.arn)

Autoruns имеет собственный двоичный файловый формат с расширением .arn. Щелкните кнопку **Save** на панели инструментов и укажите имя файла, в который хотите сохранить текущие результаты, оставив тип файлов **Autoruns Data (*.arn) [Данные Autoruns (*.arn)]**. Будет сохранена вся информация, представленная на вкладке **Everything (Все)**. Если включен

параметр **Include Empty Locations**, будут записаны все точки ASEP. Если включены параметры **Hide…**, соответствующие элементы не будут сохранены. Если предпринимались попытки проверки подписи файлов, сохранятся результаты этих проверок.

Можно автоматизировать получение данных и сохранение в файл .arn, используя параметр командной строки –a. Чтобы записать в файл outputfile.arn состояние элементов автозапуска, используя параметры Autoruns по умолчанию, выполните следующую команду:

```
Autoruns -a outputfile.arn
```

Чтобы добавить проверку подписей, укажите параметр –v, но не помещайте его между параметром –a и именем файла:

```
Autoruns -v -a outputfile.arn
```

Просмотр и сравнение результатов

Для просмотра файла .arn на том же или на другом ПК, выберите в меню **File (Файл)** команду **Open (Открыть)** и выберите сохраненный файл.

Чтобы сравнить результаты, показанные в Autoruns, будь то свежие данные или информация из сохраненного файла, выберите в меню **File (Файл)** команду **Compare (Сравнить)** и укажите сохраненный файл, с результатами из которого хотите провести сравнение. Элементы, различающиеся в двух наборах, будут выделены зеленым, как и элементы, появившиеся в первом наборе, но отсутствующие во втором. Однако удаленные элементы не отображаются. Чтобы обойти эту особенность, можно сохранить наборы как **Файл1.arn** и **Файл2.arn**, затем открыть **Файл1.arn** и сравнить его с **Файл2.arn**, после чего открыть **Файл2.arn** и сравнить его с **Файл1.arn**.

AutorunsC

AutorunsC — консольная версия Autoruns, выводящая результаты в стандартный выход и предназначенная в основном для использования в сценариях. Эта версия только собирает данные и не может отключать или удалять элементы автозапуска.

Параметры командной строки перечислены в табл. 5-11. Они позволяют получать данные обо всех элементах автозапуска или только для определенных категорий, проверять цифровые подписи, пропускать элементы Microsoft, указывать учетную запись пользователя, для которой требуется получить данные, а также сохранять результаты в виде значений, разделенных запятыми (CSV), или в виде файла XML. Если не указать никакие параметры, AutorunsC просто выведет элементы из категории **Logon (Вход в систему)** без проверки подписей в виде выровненного списка, удобного для чтения.

Независимо от формата вывода, выходные данные AutorunsC всегда включают расположение ASEP, имя элемента, описание, издателя, путь к

образу и указание на то, включен элемент или нет. Также включены хеш-строки файла образа MD5, SHA-1 и SHA-256.

В формате CSV включены заголовки столбцов. Данные в этом формате легко импортировать в Excel или реляционные СУБД. Формат XML удобен для PowerShell и других программ, работающих с данными XML. Например, следующие команды PowerShell выполняют AutorunsC, считывают файл XML и отображают отключенные элементы:

```
$arcx =  [xml]$( AutorunsC -a -x -accepteula)
$arcx.SelectNodes(«/autoruns/item»)   |   ?{  $_.enabled  -ne «Enabled»  }
```

Табл. 5-11. Параметры командной строки AutorunsC

Параметр	Описание
-c	Выходных данные записываются в формате CSV
-x	Выходные данные записываются в формате XML
-v	Проверяет цифровые подписи
-m	Скрывает элементы Microsoft
Пользователь	Указывает имя пользователя, для которого выполняется проверка элементов автозапуска
Типы элементов автозапуска	
-a	Отображаются все записи
-b	Отображаются элементы, выполняющиеся при загрузке
-d	Отображаются библиотеки DLL Appinit
-e	Отображаются дополнения к Проводнику
-g	Отображаются гаджеты боковой панели (Windows Vista и последующие версии)
-h	Отображаются краденые образы
-i	Отображаются дополнения к Internet Explorer
-k	Отображаются известные DLL
-l	Отображаются элементы автозапуска при входе в систему (применяется по умолчанию)
-n	Отображаются поставщики протокола Winsock и сети
-o	Отображаются кодеки
-p	Отображаются DLL мониторов печати
-r	Отображаются поставщики безопасности LSA
-s	Отображаются службы и драйверы
-t	Отображаются запланированные задания
-w	Отображаются элементы Winlogon

Autoruns и вредоносное ПО

Большинство вредоносных программ стремиться бесконечно оставаться на пораженной системе. Поэтому такое ПО всегда использовало точки ASEP. Сначала это были простые места, такие как раздел Run в ветви HKLM. Со временем вредоносные программы усложнялись, их становилось все труднее отследить, и способы использования ASEP тоже становились все более изощренными. Эти программы маскировались под поставщиков Winsock и мониторы печати. Такие точки ASEP не только сложно найти, но и программы, использующие их, не появляются в списке процессов, поскольку загружаются как DLL в существующие «нормальные» процессы. С ростом числа пользователей, имеющих ограниченный набор стандартных привилегий, вредоносное ПО приобретало все больше способностей к инфицированию и запуску без администраторских привилегий.

Кроме того, вредоносные программы часто используют руткиты, разрушая целостность операционной системы. Руткиты перехватывают и изменяют системные вызовы, сообщая неверную информацию о состоянии системы приложениям, использующим документированные системные интерфейсы. Руткиты могут скрывать наличие разделов и значений реестра, файлов и папок, процессов, сокетов, учетных записей пользователей и других элементов системы, и наоборот, могут заставить приложения верить, что существуют элементы, которых нет. Проще говоря, если в системе есть вредоносная программа с администраторскими привилегиями, отчетам этой системы о ее состоянии нельзя доверять. Поэтому Autoruns не всегда может идентифицировать вредоносные элементы автозапуска.

Однако не все вредоносные программы настолько сложны, и все равно остаются некоторые признаки, выдающие его присутствие:

- элементы, для которых указан известный издатель, такой как Microsoft, не проходят проверку подписей (к сожалению, не все ПО от Microsoft подписано);
- элементы, путь которых указывает на DLL- или EXE-файл без описания или информации об издателе (кроме случаев, когда целевой файл не найден);
- обычные компоненты Windows запускаются из необычных мест; например, если svchost.exe запускается из C:\Windows или C:\System Volume Information, а не из System32;
- элементы, для которых временная отметка файла запущенной программы совпадают с временем первого проявления проблем или обнаружения атаки;
- элементы, которые после отключения или удаления и обновления экрана восстанавливаются и включаются. Вредоносное ПО часто отслеживает свои точки ASEP и восстанавливает их в случае удаления.

Вредоносные программы и средства для борьбы с ними все время эволюционируют. Сегодняшние рекомендации завтра могут показаться наивными и неэффективными.

Есть и элементы, которые могут показаться подозрительными, но на самом деле они безвредны. Вот они:

- в стандартной установке Windows XP или Windows Server 2003 будет показано несколько элементов **File not found (Файл не найден)**, в частности, на вкладке **Drivers (Драйверы)**. Это устаревшие элементы, файлы которых были удалены из дистрибутива, но элементы реестра по невнимательности не очищены;
- в стандартной установке Windows XP на вкладке **Image Hijacks** в разделе **Image File Execution Options (IFEO)** будет показан элемент **Your Image File Name Here without a path (Имя файла образа без пути)**, указывающий на файл Ntsd.exe из папки System32. Этот элемент служит примером использования раздела IFEO;
- в стандартной установке Windows Vista на вкладке **Drivers (Драйверы)** может отображаться несколько элементов **File not found** для драйверов NetWareIPX и для драйвера «IP in IP Tunnel Driver»;
- в стандартной установке Windows 7 на вкладке **Scheduled Tasks (Запланированные задания)** в разделе **\Microsoft\Windows** может иметься несколько элементов, содержащих только имя элемента и ничего больше.

Глава 6

PsTools

PsTools от Sysinternals — набор из 12 утилит для управления Windows с общими характеристиками:

- все они являются консольными, т.е. предназначены для запуска из командной строки или пакетного файла и выводят информацию в стандартные потоки вывода и сообщений об ошибках (которые могут отображаться в окне консоли или перенаправляться в файлы);
- работают на локальном или удаленном компьютере. В отличие от большинства программ для удаленного управления, утилиты PsTools не требуют предварительной установки клиентского ПО на удаленных системах (разумеется, как и все утилиты Sysinternals, установки на локальном компьютере они тоже не требуют);
- они поддерживают стандартный синтаксис для указания альтернативных учетных данных, так что все задачи можно выполнять от имени другого пользователя.[1]

В пакет PsTools входят следующие утилиты:

- **PsExec** — выполняет процессы удаленно и (или) от имени локальной системы, перенаправляя выходные данные;
- **PsFile** — отображает список файлов, открытых удаленно, или закрывает их;
- **PsGetSid** — отображает идентификатор защиты (SID) компьютера, пользователя или группы;
- **PsInfo** — отображает информацию о системе;
- **PsKill** — завершает процессы, заданные по имени или по идентификатору процесса (PID);
- **PsList** — отображает информацию о процессах;
- **PsLoggedOn** — перечисляет учетные записи, под которыми пользователи вошли в систему локально и удаленно;

---

[1] Есть два исключения: альтернативные учетные данные не принимает PsLoggedOn, а также PsPasswd при изменении пароля для доменной учетной записи.

- **PsLogList** — выгружает записи из журнала событий;
- **PsPasswd** — изменяет пароли учетных записей пользователей;
- **PsService** — отображает список служб Windows и управляет ими;
- **PsShutdown** — выключает компьютер, выходит из системы или изменяет состояние питания локальной или удаленной системы;
- **PsSuspend** — приостанавливает и возобновляет работу процессов.

Кстати, этот набор называется PsTools, а все утилиты имеют префикс Ps в названии, потому что первую из этих утилит, PsList, отображающую список запущенных процессов, я назвал в честь *ps*, UNIX-утилиты со сходной функциональностью.

Прежде чем мы начнем разбираться с утилитами, хочу прояснить одну вещь. Бывает, что антивирусы сообщают о том, что некоторые из утилит PsTools являются троянскими программами или вредоносным ПО другого типа. Уверяю вас, что *никакие* из утилит PsTools и Sysinternals вообще не являются вредоносными. Однако злоумышленники встраивают различные утилиты PsTools, в частности, PsExec, в свои вредоносные программы. Так как в PsTools указано мое имя и веб-сайт, а авторы вредоносных программ обычно не указывают свои контакты, все письма от рассерженных пользователей, обвиняющих меня в написании вирусов и заражении их систем, приходят мне. Как я уже много раз объяснял, утилиты PsTools созданы с благими намерениями, а с тем, что их используют во вред, я ничего не могу поделать. Более того, эти утилиты не используют никаких уязвимостей и не получают несанкционированного доступа. Они должны запускаться под учетной записью, уже имеющей необходимые права доступа, или им необходимо предоставлять имя и пароль соответствующей учетной записи.

Общие функции

Все утилиты PsTools работают во всех поддерживаемых клиентских и серверных версиях Windows, к которым относятся 32- и 64-разрядные редакции Windows XP, Windows Vista и Windows 7, а также редакции x86, x64 и IA-64 Windows Server 2003, 2008 и 2008 R2. Для поддержки 64-разрядных версий требуется установить компоненты WOW64, обеспечивающие поддержку 32-разрядных приложений в 64-разрядных версиях Windows (в редакции Server Core WOW64 можно удалить).

Все утилиты PsTools поддерживают удаленное выполнение операций с использованием синтаксиса, общего для всего пакета. Чтобы посмотреть синтаксис вызова утилиты, запустите ее из командной строки с параметром `-?`. Синтаксис командной строки всех утилит PsTools приведен в разделе «Синтаксис командной строки PsTools» ближе к концу этой главы.

Удаленные операции

Утилиты PsTools могут выполнять операции на локальном или удаленном компьютере. Каждая из этих утилит принимает необязательный параметр

командной строки вида \\имя_компьютера либо \\IP-адрес_компьютера, на котором должны выполняться операции. Например:

```
psinfo \\srv2008r2
psinfo \\192.168.0.10
```

Некоторые из утилит для выполнения удаленных операций просто используют интерфейсы Windows API, позволяющие указывать удаленный компьютер. Другие извлекают для этого EXE-файл, внедренный в их исполняемый образ, копируют этот файл в общую папку Admin$ удаленного компьютера, регистрируют его на той системе как службу и запускают эту службу при помощи интерфейсов API Windows Service Control Manager, после чего взаимодействуют с ней по именованным каналам. Для создания удаленной службы требуется, чтобы на целевом компьютере был включен общий доступ к файлам и открыт доступ к папке Admin$. В конце главы приведена таблица со списком утилит PsTools, требующих включения этих функций для удаленной работы.

Удаленные операции на нескольких компьютерах

Некоторые утилиты можно запускать на нескольких удаленных компьютерах одной командой (в таблице в конце главы указано, какие утилиты поддерживают такую возможность). Для этих утилит можно указать удаленные компьютеры непосредственно в командной строке или во входном файле. Синтаксис командной строки следующий: два обратных слеша, за которыми идут имена или IP-адреса компьютеров, разделенные запятыми без пробелов. Например:

```
psinfo \\server1, server2,192.168.0.3,server4
```

Эта команда выводит информацию о системе **server1**, затем о системе **server2**, затем о системе с IP-адресом **192.168.0.3** и, наконец, о системе **server4**.

Другой способ указать удаленные компьютеры — использовать текстовый файл, содержащий имя или IP-адрес каждого компьютера в отдельной строке. Имя файла указывается в командной строке с префиксом @. Если предыдущий пример выполнять с использованием файла *computers.txt*, то этот файл должен содержать следующие строки:

```
server1
server2
192.168.0.3
server4
```

А команда будет выглядеть так:

```
psinfo @computers.txt
```

Если утилите, которую можно выполнять на нескольких удаленных компьютерах одновременно, передать параметр командной строки *,

она будет выполнена на всех компьютерах текущего домена или рабочей группы:

```
psinfo \\*
```

Если не использовать ни один из этих параметров, утилита будет выполнять операции на локальном компьютере.

Альтернативные учетные данные

При выполнении операций на удаленных компьютерах утилиты PsTools действуют от имени учетной записи, под которой они запущены на локальной системе. Если это локальная, а не доменная учетная запись домена, аутентификация пройдет, только если на удаленном компьютере имеется учетная запись с таким же именем и паролем.

Есть несколько причин для запуска утилит от имени другой учетной записи на удаленной системе. Во-первых, большинство утилит требуют административных привилегий на целевом компьютере, так что потребуется учетная запись, имеющая такие привилегии. Во-вторых, как будет говориться далее, в Windows Vista появились ограничения на использование локальных учетных записей для удаленного администрирования. Наконец, есть несколько причин, относящихся только к PsExec, о которых будет говориться в разделе, посвященном этой утилите.

Чтобы использовать другую учетную запись, укажите ее с параметром командной строки –u, и если требуется, укажите пароль с параметром –p. Например:

```
psinfo \\server1 -u ИМЯДОМЕНА\AdminAccnt -p Pass@word123
```

Если имя пользователя или пароль содержат пробелы, заключайте их в двойные кавычки:

```
psinfo \\server1 -u "ИМЯДОМЕНА\Имя Пользователя" -p "Пароль с пробелами"
```

Если опустить параметр –p, утилита предложит ввести пароль. По соображениям безопасности символы пароля не видны на экране при вводе с клавиатуры. Утилиты используют API WNetAddConnectoin2, так что пароли не пересылаются по сети открытым текстом. (Однако в некоторых случаях утилите PsExec приходится отправлять учетные данные уже после аутентификации для создания нового сеанса; в этом случае учетные данные отправляются без шифрования.)

Параметры –u и –p поддерживаются всеми утилитами PsTools, кроме PsLoggedOn.

Решение проблем с удаленными подключениями PsTools

Для работы утилит PsTools на удаленных системах должны выполняться некоторые условия. Очевидно, что требуется возможность подключения к необходимым сетевым интерфейсам, для чего может понадобиться настрой-

ка соответствующих параметров брандмауэра и запуск служб. Большинство утилит требует администраторских прав. Наконец, следует учитывать, что средства контроля учетных записей пользователей (User Account Control, UAC) ограничивают использование локальных учетных записей.

Базовое подключение

Если не указывается IP-адрес, необходимы средства разрешение имен. Если служба DNS недоступна, можно использовать NetBIOS через TCP (NBT), но для этого на брандмауэре целевой системы должны быть открыты порты 137 UDP, 137 TCP, 138 UDP и 139 TCP.

Некоторым утилитам необходим доступ к общей папке Admin$. Для этого требуется включение совместного использования файлов и принтеров (служба Workstation [Рабочая станция] на локальной системе и служба Server [Сервер] на целевой). Кроме того, брандмауэр не должен блокировать порты, необходимые для поддержки общего доступа к файлам и принтерам, а «простой общий доступ к файлам» должен быть отключен.

Некоторые утилиты требуют запуска службы Remote Registry (Удаленный реестр) на целевой системе. (В таблице в конце главы указано, какие утилиты требуют этого.) Обратите внимание, что в последних версиях Windows эта служба не запускается автоматически по умолчанию. Поэтому ее следует запускать вручную или настроить ее автоматический запуск, прежде чем запускать утилиты, работающие с ней.

Учетные записи пользователей

Большинство утилит требует прав администратора. До выпуска Windows Vista и появления UAC с административными учетными записями было все просто. Если учетная запись входила в группу **Администраторы (Administrators)**, все, что запускалось от имени этой учетной записи, получало администраторские права. После успешной аутентификации под учетной записью из группы администраторов предоставлялся полный доступ к системе.

В Windows Vista появился контроль учетных записей пользователей (UAC), в котором, кроме всего прочего, была реализована концепция учетных записей, одновременно являющихся и административными, и обычными пользовательскими. Такие учетные записи иногда называют *защищенными администраторскими*. Идея заключается в том, что программы запускаются с привилегиями обычного пользователя, а если программа требует полных прав администратора, пользователь должен явно разрешить повышение привилегий. Программы, запускаемые от имени пользователя, не должны программно подтверждать повышение или другим способом обходить взаимодействие с пользователем. Если бы они могли это, разработчики применяли бы такие обходные пути и продолжали писать программы, требующие прав администратора, а не обычного пользователя.

Интерфейс «обратная петля» (loopback) — один из путей автоматического повышения прав, который блокируется в Windows Vista. Как описано в статье 951016 базы знаний Knowledge Base, если устанавливается сетевое подключение к удаленному компьютеру с использованием локальной учетной записи из группы администраторов, предоставляются только привилегии обычного пользователя. Так как это не интерактивный вход, нет возможности повысить права до полных административных. К учетным записям домена это ограничение не применяется.

Это значит, что, хотя утилиты PsTools замечательно подходят для удаленного администрирования Windows XP и Windows Server 2003 с использованием локальных учетных записей, такой подход не очень хорошо работает в Windows Vista и последующих версиях. Если нельзя использовать доменные учетные записи, прочитайте статью KB 951016, в которой рассказывается, как настроить параметр LocalAccountTokenFilterPolicy, чтобы убрать ограничения на использование локальных учетных записей.

PsExec

Утилита PsExec позволяет выполнять произвольные процессы на одном или нескольких удаленных компьютерах. PsExec перенаправляет потоки ввода-вывода консольных приложений, так что они выглядят как запущенные через сеанс Telnet. Таким образом, консольные утилиты, обычно работающие только на локальном компьютере, получают возможность удаленного выполнения. В частности, эта возможность позволяет запускать оболочку командной строки на удаленной системе и работать с ней как с локальной. В отличие от большинства средств удаленного управления, PsExec не требует предварительной установки агентов или другого клиентского ПО на целевом компьютере. Конечно, требуется учетная запись, авторизованная для удаленного администрирования компьютера.

PsExec также позволяет запускать программы локально или удаленно от имени учетной записи системы, как интерактивно, так и не интерактивно. Например, можно запустить Regedit и просмотреть иерархию доступных только учетной записи SYSTEM разделов реестра, таких как HKLM\SAM и HKLM\Security. Как говорилось в главе 4, PsExec может запустить программу в неинтерактивном сеансе, который останется активным после выхода пользователя из системы. PsExec предоставляет много других возможностей по управлению запуском локального или удаленного целевого процесса, включая указание учетной записи, уровня привилегий, приоритета и привязки к ЦП.

Команда для запуска процесса на удаленном компьютере имеет следующий синтаксис:

```
psexec \\компьютер [параметры] программа [аргументы]
```

Например, чтобы запустить на удаленной системе команду ipconfig /all и просмотреть ее выходные данные на локальном компьютере, выполните следующую команду:

```
psexec \\server1 ipconfig /all
```

Чтобы запустить процесс на локальном компьютере, просто опустите параметр \\computer:

```
psexec [параметры] программа [аргументы]
```

Если имя программы (часть программа) содержит пробелы, необходимо заключить путь к программе в кавычки. Если удаленная командная строка содержит специальные знаки, такие как символы канала или символы перенаправления, используйте управляющий символ оболочки командной строки (^), чтобы локальная оболочка командной строки не обрабатывала эти специальные символы. Приведем пример выполнения команды ipconfig /all на компьютере server1 с перенаправлением стандартного выхода в файл **C:\ipconfig.out** на целевой системе:

```
psexec \\server1 cmd.exe /c ipconfig /all  ^> c:\ipconfig.out
```

Без управляющего символа (^) стандартный выход команды PsExec (включая перенаправленный выход команды ipconfig) будет направлен в файл **c:\ipconfig.out** на локальном компьютере. (Диагностические данные PsExec записываются в стандартный поток вывода сообщений об ошибках, а не стандартный выходной поток, так что при перенаправлении на локальную систему захватываются только выходные данные удаленного процесса.)

Если в командной строке PsExec указано только имя файла (в части программа), этот файл будет записан на удаленной системе по пути, заданному параметром Path. (Обратите внимание: изменения глобальной переменной окружения PATH не видны службам до следующей перезагрузки.) Если аргумент программа указывает абсолютный путь, буквы дисков должны соответствовать таковым на удаленной системе. Например, C: будет указывать на диск C: удаленной системы, а буквы сетевых дисков на локальном компьютере или буквы дисков, подключенных во время пользовательского сеанса, не будут распознаваться. Если нужной программы еще нет на удаленной системе, PsExec может скопировать ее с локального компьютера на удаленный (см. раздел «Параметры удаленного подключения» далее в этой главе).

Завершение удаленного процесса

По умолчанию PsExec не завершает работу, пока не завершена работа программы, запущенной этой утилитой. Когда процесс завершает работу, он сообщает операционной системе Код завершения, 32-разрядное целое число, которое может быть считано родительским процессом (или любым другим процессом, имеющим открытый описатель завершившегося процесса). Код

завершения часто используют, чтобы сообщить, успешно ли процесс выполнил свою задачу. Ноль обычно указывает на успешное выполнение. Код завершения проверяется командой IF ERRORLEVEL оболочки cmd и ее условными операторами && и ||. PsExec выводит Код завершения процесса в консоль (например, «Notepad.exe exited with error code 0»). После этого PsExec завершает свою работу, указывая код завершения целевой программы в качестве собственного кода. Так родительскому процессу или пакетному файлу удается проверить, успешно ли он завершился, и выполнить необходимые действия по этому условию.

Когда используется параметр –d, PsExec запускает удаленный процесс, но не ждет его завершения. При успешном запуске PsExec выводит идентификатор (PID) нового процесса в поток *stderr* и завершается, используя этот идентификатор в качестве кода завершения. В пакетном файле этот идентификатор можно получить следующим образом:

```
psexec \\server1 -d App.exe
SET NEWPID=%ERRORLEVEL%
ECHO The Process ID for App.exe is %NEWPID%
```

Если же PsExec не может запустить удаленный процесс, код завершения является кодом ошибки. Выяснить, является ли код завершения идентификатором процесса или кодом ошибки, программным способом нельзя.

Перенаправление вывода консоли

Чтобы запустить оболочку командной строки на удаленной системе и работать с ней на локальном компьютере, просто выполните команду следующего вида:

```
psexec \\server1 Cmd.exe
```

При перенаправлении вывода консоли учитывайте следующее:

- операции, требующие знания содержимого экрана консоли, такого как позиция курсора или цвет текста, не работают. Сюда входят команды cls (очистка экрана) и more, а также табуляция имен файлов и папок;
- если запустить программу в новом окне, например, использовать команду start или запустить любую программу с графическим интерфейсом, эта программа будет запущена на удаленном компьютере, но с ней нельзя будет взаимодействовать;
- все утилиты Sysinternals, включая консольные утилиты, отображают диалоговое окно с лицензионным соглашением при первом запуске под новой учетной записью, если не добавить параметр командной строки /accepteula. С учетом того, что написано в предыдущем пункте, это диалоговое окно нельзя будет закрыть, а утилита будет ждать принятия соглашения, пока не будет закрыта нажатием Ctrl+C. При перенаправлении выходных данных утилит Sysinternals используйте параметр /accepteula;

Примечание Некоторые утилиты Sysinternals еще не поддерживают параметр /accepteula. Для таких утилит необходимо вручную устанавливать флаг принятия соглашения. Это можно сделать следующей командой:

```
psexec \\server1 reg add hkcu\software\sysinternals\pendmove /v
eulaaccepted /t reg_dword /d 1 /f
```

- Windows PowerShell версии 1 не поддерживает перенаправление вывода консоли, но вторая версия его поддерживает, если запускать ее с параметром командной строки –File –. Например:

```
psexec \\server1 PowerShell.exe -file -
```

- нажатие Ctrl+C принудительно завершает удаленный процесс, а не только текущую команду. Например, если вы запустили удаленную оболочку командной строки и случайно выполнили команду dir /s c:\, нажатие Ctrl+C закроет оболочку, а не только завершит команду dir.

Некоторые распространенные команды, такие как dir и copy, не являются отдельными исполняемыми программами, а встроены в Cmd.exe. Чтобы выполнить встроенную команду, используйте параметр /c оболочки Cmd для выполнения этой команды в контексте процесса Cmd.exe, который будет завершен после выполнения команды. Например, следующая команда запускает на компьютере server1 экземпляр Cmd.exe, выполняющий встроенную команду ver, а затем закрывающийся:

```
psexec \\server1 Cmd.exe /c ver
```

Выходные данные команды `ver`, выполненной на server1, отображаются в локальном окне консоли, из которого запущена утилита PsExec. В этом случае Cmd.exe входит в путь (параметр `программа` командной строки PsExec), а `/c ver` является необязательной частью параметра `аргументы`, передаваемого запускаемой программе.

Альтернативные учетные данные PsExec

В разделе «Альтернативные учетные данные» выше описано использование параметров –u и –p для явного предоставления учетных данных утилитам PsTools. Если эти параметры не указаны, для аутентификации на удаленной системе будет использована учетная запись пользователя, запустившего PsExec, и удаленный процесс, запущенный PsExec, будет действовать от имени этой учетной записи. При этом возникают следующие ограничения:

- чтобы запустить процесс на удаленной системе, PsExec необходимо использовать учетную запись, имеющую там администраторские права;
- если удаленный процесс обращается к сетевым ресурсам, он будет проходить анонимную аутентификацию, если только не включено делегирование Kerberos. Это так называемое *ограничение одного прыжка*: компьютер, на котором вход в систему выполнен с явным указанием учетных данных, может пройти аутентификацию на удаленном сервере, который

может действовать в этом контексте безопасности, но процесс на удаленном компьютере не может после этого использовать этот контекст безопасности для аутентификации на третьей системе;

- полученный контекст безопасности не будет включать SID входа, открывающие доступ к интерактивным сеансам пользователя.

Следует явно указывать учетные данные, если учетная запись, от имени которой запускается PsExec, не имеет прав администратора на удаленном компьютере, если удаленный процесс требует аутентифицированного доступа к сетевым ресурсам, или если удаленный процесс должен выполняться на интерактивном рабочем столе пользователя. Когда учетные данные указываются явно, они используются для прохождения аутентификации на удаленной системе, а затем для создания нового сеанса входа, который можно запускать на интерактивном рабочем столе.

Внимание! Имя пользователя и пароль для создания нового сеанса входа передаются удаленной системе открытым текстом, без шифрования. Любой человек, прослушивающий сеть, сможет перехватить эти учетные данные. Для защиты шифрованием всех передаваемых данных можно использовать IPSec с ESP (Encapsulating Security Payload).

Параметры –u и –p можно использовать также при запуске процесса на локальном компьютере, аналогично RunAs.exe. Как и при запуске при помощи RunAs.exe, из-за UAC целевой процесс не будет иметь полных прав администратора в Windows Vista и последующих версиях, даже если учетная запись входит в группу администраторов (если только не использовать параметр –h, описанный ниже).

Параметры командной строки PsExec

Рассмотрим параметры командной строки PsExec. Они управляют различными аспектами производительности процессов, удаленного подключения и исполняющей среды, а также определяют, будет ли PsExec ожидать завершения целевого процесса. Эти параметры вкратце описаны в табл. 6-1, а подробнее будут рассматриваться далее.

Табл. 6-1. Параметры командной строки PsExec

Параметр	Описание
-d	PsExec не ожидает завершения процесса (см. раздел «Завершение удаленного процесса» выше)
Параметры производительности процесса	
-background -low -belownormal -abovenormal -high -realtime	Запускают процесс с указанным приоритетом
-a n,n...	Указывает процессоры, которые может использовать процесс

Табл. 6-1. (*окончание*)

Параметр	Описание
Параметры удаленного подключения	
-c [-f | -v]	Копирует указанную программу из локальной системы на удаленную. Если опустить этот параметр, приложение должно иметься в системном пути на удаленном компьютере. При добавлении параметра –f копирование выполняется принудительно. Если указать параметр –v, будет проверена версия или временной штамп, и копирование будет выполнено только в том случае, если источник новее
-n *секунды*	Указывает таймаут в секундах при подключении к удаленному компьютеру
Параметры исполняющей среды	
-s	Выполняет процесс от имени учетной записи SYSTEM
-i [*сеанс*]	Выполняет программу на интерактивном рабочем столе
-x	Выполняет процесс на безопасном рабочем столе Winlogon
-w *каталог*	Определяет рабочий каталог процесса
-e	Не загружает профиль указанной учетной записи
-h	Использует повышенный контекст учетной записи, если тот доступен
-l	Выполняет процесс от имени ограниченной учетной записи

Параметры производительности процесса

По умолчанию целевой процесс запускается с обычным приоритетом. Можно явно указать приоритет процесса, используя следующие параметры PsExec: –background, –low, -belownormal, –abovenormal, –high и –realtime. Параметр –background поддерживается, начиная с Windows Vista; в дополнение к назначению процессу низкого (Low) приоритета, он устанавливает очень низкие (Very Low) приоритеты обращения к памяти и ввода-вывода. Если целевая система является многопроцессорной, можно указать, что целевой процесс будет работать только на определенных процессорах. Для этого добавьте параметр –a со списком логических ЦП, разделенных запятыми (где 1 — процессор с наименьшим номером). Например, чтобы процесс использовал только третий процессор, выполните следующую команду:

```
psexec -a 3 app.exe
```

Чтобы запустить целевой процесс на процессорах с номерами 2, 3 и 4, выполните следующую команду:

```
psexec -a 2,3,4 app.exe
```

Параметры удаленного подключения

Если программа, которую требуется запустить, не установлена на удаленной системе, PsExec может скопировать ее из локальной файловой системы в папку system32 удаленного компьютера, запустить ее оттуда, а затем удалить после выполнения. Можно добавить условие, что программа будет копироваться только в том случае, если на удаленной системе нет ее более новой версии. Если указать параметр –c, часть программа командной строки PsExec указывает путь к файлу относительно локального компьютера. Этот файл будет скопирован в папку system32 удаленной системы. Заметьте, однако, что копируется только указанный файл, но не связанные библиотеки DLL или другие файлы.

При использовании параметра –c копирование не будет выполнено, если такой файл уже есть в целевой папке. При добавлении параметра –f копирование выполняется принудительно, даже если требуется переписать файл, помеченный как доступный только для чтения, скрытый или системный. При добавлении параметра –v выполняется проверка версий и временных штампов файлов. Файл копируется, только если это более поздняя версия или более новый файл. Удаленный процесс запускается в любом случае.

При попытке установления подключения к удаленной системе, которая выключена, очень загружена или имеет проблемы с подключением, PsExec использует стандартный системный тайм-аут для каждой из необходимых сетевых операций. Чтобы указать более короткий период ожидания, используйте параметр –n и укажите максимальное число секунд ожидания каждого удаленного подключения. Например, чтобы ограничить время на попытку подключения к группе удаленных систем десятью секундами, используйте следующую команду:

```
psexec @computers.txt -n 10 app.exe
```

Параметры исполняющей среды

PsExec предоставляет несколько параметров командной строки для управления исполняющей средой целевого процесса. Эти параметры позволяют выполнять процесс от имени учетной записи системы и учетной записи с ограниченными привилегиями, запускать процесс в интерактивном режиме и выбирать интерактивный сеанс, указывать, нужно ли загружать профиль учетной записи на целевую систему, и указывать исходный рабочий каталог для целевого процесса.

Параметр –s запускает целевое приложение от имени учетной записи SYSTEM. Если не указать также параметр –i (см. ниже), процесс будет запущен в той же неинтерактивной среде, в которой работают остальные службы Windows, запущенные от имени системы (сеанс 0, оконная станция Service-0x0-3e7$; подробнее см. в гл. 2), а вывод консоли будет перенаправлен в ту консоль, в которой работает PsExec. О том, что при этом следует учитывать,

рассказано выше в разделе «Перенаправление вывода консоли». Одним из преимуществ такого режима выполнения является то, что процесс продолжает работу даже после выхода из системы интерактивного пользователя. В главе, посвященной Process Monitor, приведен пример такого использования PsExec для мониторинга событий во время выхода пользователя из системы и завершения работы.

Если целевой системой является локальный компьютер, утилита PsExec должна работать с полными административными правами, чтобы можно было использовать параметр –s. Для удаленного выполнения PsExec требует запуска от имени администраторской учетной записи на удаленной системе.

Параметр –i [сеанс] используется для интерактивного запуска целевого процесса на целевой системе, а именно, на выбранном по умолчанию интерактивном рабочем столе сеанса служб терминалов. Без параметра –i процессы на удаленных компьютерах будут запускаться в неинтерактивной оконной станции в сеансе 0. Необязательный параметр сеанс указывает идентификатор сеанса служб терминалов, в котором должен работать процесс. Если использовать параметр –i, но опустить параметр сеанс, PsExec запустит процесс в текущем сеансе рабочего стола при запуске на локальном компьютере, или в текущем консольном сеансе при запуске на удаленном компьютере. Консольный сеанс — это текущий сеанс, связанный с клавиатурой и монитором, подключенным к компьютеру (в противоположность сеансу удаленного рабочего стола). Напомним, что для запуска интерактивного процесса на удаленном компьютере требуется явно указать учетные данные.

Совет Чтобы просмотреть идентификаторы сеансов, связанные с процессами, включите отображение столбца **Session (Сеанс)** в Process Explorer (см. гл. 3).

Следующая команда запускает Regedit от имени учетной записи SYSTEM в текущем интерактивном сеансе, что позволяет просмотреть те части реестра, доступ к которым разрешен только системе (такие как HKLM\SAM и HKLM\Security):

```
psexec -s -i Regedit.exe
```

А эта команда запускает от имени системы оболочку командной строки на текущем рабочем столе:

```
psexec -s -i Cmd.exe
```

Параметр –x запускает целевой процесс на защищенном рабочем столе Winlogon. Рабочий стол Winlogon управляется учетной записью системы, и доступ к нему могут получать только процессы, работающие от ее имени. Обычно это значит, что –x требуется использовать вместе с –s, а утилита PsExec уже должна работать с административными правами. Кроме того, параметр –x можно использовать только на локальном компьютере. По умолчанию –x запускает целевой процесс на рабочем столе Winlogon консольного

сеанса. Используйте параметр –i вместе с –x, чтобы запустить целевой процесс на рабочем столе Winlogon в другом сеансе удаленного рабочего стола. Следующая команда запускает оболочку командной строки на безопасном рабочем столе консольного сеанса:

```
psexec -x -s Cmd.exe
```

Если вы вошли в систему в консольном режиме, нажмите Ctrl+Alt+Del, чтобы переключиться на рабочий стол Winlogon. Если в вашей версии Windows отображается полноэкранное изображение безопасного рабочего стола, нажмите Alt+Tab, чтобы перейти в оболочку командной строки.

Параметр –w каталог устанавливает исходный каталог целевого процесса. Заметьте, что путь к каталогу указывается относительно целевого компьютера. Например, путь **C:\Program Files** ссылается на папку **C:\Program Files** удаленного компьютера, а не локального. Также обратите внимание на то, что буквы, назначенные сетевым дискам, обычно не распознаются.

Если используется параметр –e, профиль учетной записи пользователя не загружается. Эта функция может сэкономить немного времени для процессов с коротким временем жизни, которым профиль учетной записи не нужен. Однако ее не следует использовать в тех случаях, когда выполнение операций может зависеть от параметров профиля пользователя. Ветвь HKCU, просматриваемая процессом, ссылается на ветвь HKCU учетной записи системы, если только в другом сеансе профиль пользователя уже не был загружен к тому времени, когда был запущен процесс. Переменная окружения %USERPROFILE% ссылается на каталог профиля учетной записи системы независимо от того, был ли загружен профиль пользователя. Так как профиль учетной записи системы загружается всегда, PsExec не позволяет использовать параметры –e и –s одновременно.

В Windows Vista и последующих версиях для входа «интерактивного» типа (когда явно указываются учетные данные) действует фильтрация маркеров: административные группы отключаются, а административные привилегии удаляются. Когда учетные данные указываются явно, добавление параметра –h запускает целевой процесс на удаленной системе с полным маркером администраторской записи. Если целевой системой является локальный компьютер, параметр –h запускает процесс с повышенным маркером только в том случае, если PsExec уже работает с повышенными правами.

Параметр –l (L в нижнем регистре) запускает целевой процесс с ограниченными правами. Если в маркере пользователя представлена группа администраторов, она отключается; также удаляются все привилегии, кроме предоставленных группе **Пользователи (Users)** на целевом компьютере. В Windows Vista и последующих версиях процесс запускается с низким (Low) уровнем целостности (IL), что запрещает ему вести запись в большинство мест файловой системы и реестра. Следующая команда позволяет администратору Windows XP запустить Internet Explorer с ограниченными правами:

```
psexec -l -d  «%ProgramFiles%\Internet Explorer\iexplore.exe»
```

Примечание Полученный процесс с «ограниченными правами» не обязательно будет иметь те же характеристики, что и другие процессы с «низкими правами», работающие на компьютерах с Windows, такие как Internet Explorer в защищенном режиме. PsExec не отключает группы с повышенными привилегиями, которые обычно отключает UAC (такие как **Опытные пользователи [Power Users]** и другие доменные группы), кроме группы **Администраторы (Administrators)**. Кроме того, если процесс запущен из процесса с повышенными правами, маркер нового процесса все равно будет получен из «повышенного» сеанса, даже если он помечен как имеющий низкий IL. Оболочка командной строки с таким маркером будет иметь на панели заголовка надпись **Администратор (Administrator)**, а дочерние процессы, требующие повышения прав, не смогут запросить или получить это повышение.

PsFile

Команда Windows NET FILE показывает список файлов, которые процессы с других компьютеров открыли в системе, на которой выполнена эта команда. Однако она обрезает длинные пути и не позволяет просматривать эту информацию для удаленных систем. PsFile показывает список файлов или именованных каналов системы, которые открыты удаленно через службу Server, а также позволяет закрывать файлы, открытые удаленно, по имени или номеру идентификатора.

По умолчаниюPsFile перечисляет файлы локальной системы, открытые с удаленных систем. Чтобы просмотреть файлы, открытые на удаленной системе, укажите имя удаленного компьютера (при необходимости предоставив альтернативные учетные данные), используя синтаксис, описанный в разделе «Общие функции» в начале главы. Выходные данные будут выглядеть примерно так:

```
Files opened  remotely on win7_vm:
[332]  C:\Users
    User:   ABBY
    Locks:  0
    Access: Read
[340] C:\Windows\TEMP\listing.txt
    User:   ABBY
    Locks:  0
    Access: Read Write
[352] \PIPE\srvsvc
    User:   ABBY
    Locks:  0
    Access: Read Write
```

Число в скобках — идентификатор, предоставленный системой, за которым следует путь к открытому файлу и имя учетной записи, связанной с удаленным подключением. В списке файлов, открытых на удаленном компьютере, всегда будет именованный канал *srvsvc*, потому что PsFile устанавливает подключение к службе Server.

Выходные данные можно фильтровать, добавив в командную строку номер идентификатора ресурса или путь. Следующая команда показывает только те сведения, которые относятся к ресурсу с идентификатором 340 на компьютере с именем Win7_vm:

```
psfile \\Win7_vm  340
```

Следующая команда показывает только сведения, относящиеся к открытым файлам из папки C:\Users, т. е. все ресурсы, пути которых начинаются с C:\Users:

```
psfile \\Win7_vm C:\Users
```

Чтобы закрыть открытые файлы, добавьте в командную строку параметр –c после идентификатора или префикса пути. Следующая команда закрывает все файлы, открытые удаленно из папки C:\Users на локальном компьютере:

```
psfile C:\Users -c
```

Используя PsFile, закрывайте файлы с осторожностью, так как данные, кешированные на клиентской системе, не записываются в файл перед его закрытием.

PsGetSid

В Windows идентификаторы защиты (SID) уникальны для пользователей, группы, компьютеров и других сущности. Эти идентификаторы хранятся в маркерах доступа и дескрипторах защиты и используются при проверке прав доступа. Имена, связанные с SID, нужны только для представления в пользовательском интерфейсе и могут изменяться на разных системах в результате локализации. Например, в англоязычных системах имеется группа Administrators, имеющая SID S-1-5-32-544, но в системах с интерфейсом на русском языке она называется Администраторы, на итальянском — Gruppo Administrators, а на финском — Jarjestelmanvalvojat.

Каждый компьютер под управлением Windows имеет локальный SID, называемый также *машинным SID*, создающийся во время установки ОС. Каждая локальная группа и учетная запись на компьютере имеет SID, созданный из машинного SID с добавлением относительного идентификатора (Relative ID, RID). Точно так же, каждый домен Active Directory имеет SID, а сущности внутри этого домена (включая доменные группы, учетные записи пользователей и компьютеров) имеют идентификаторы SID, полученные на основе SID домена с добавлением RID. Кроме машинных и доменных SID, в Windows определен набор хорошо известных идентификаторов SID в доменах NT AUTHORITY и BULTIN.

PsGetSid облегчает перевод идентификаторов SID в соответствующие имена и имен в идентификаторы SID, а также получение идентификаторов компьютеров и доменов. Как и другие утилиты PsTools, PsGetSid может вы-

полнять эти переводы на удаленных системах и сообщать результаты на локальном компьютере.

Чтобы перевести имя или SID, запустите PsGetSid с именем или идентификатором SID в командной строке. При запуске без параметров эта утилита отображает машинный SID локального компьютера. Например:

```
C:\>psgetsid
SID for \\WIN_VM:
S-1-5-21-2292904206-3342264711-2075022165

C:\>psgetsid Administrator
SID for WIN_VM\Administrator:
S-1-5-21-2292904206-3342264711-2075022165-500
```

Использование полных доменных имен (ДОМЕН\ИМЯПОЛЬЗОВАТЕЛЯ) позволяет избежать путаницы и повышает производительность. Если указать только имя учетной записи, PsGetSid в первую очередь проверяет общеизвестные SID, затем встроенные и административно определенные учетные записи. Если имя не найдено, выполняется проверка первичного домена, а затем доверенных доменов.

Перевод идентификаторов SID входа (Logon SID) невозможен. Эти идентификаторы генерируются случайным образом, они назначаются временным объектам и имеют формат S-1-5-5-X-Y.

Следующая строка сценария PowerShell возвращает список имен, связанных со стандартными SID в диапазоне от S-1-5-32-544 до S-1-5-32-576, перенаправляя любые сведения об ошибках в *nul*. Выходные данные этой команды показаны на рис. 6-1.

```
0x220..0x240   |  %{  psgetsid  S-1-5-32-$_ 2> $nul   }
```

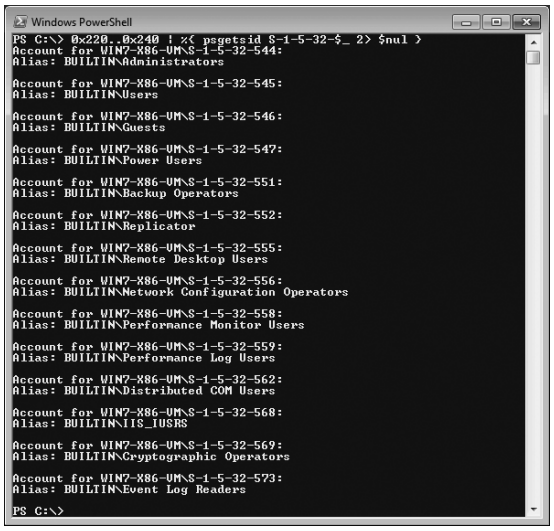


Рис. 6-1. Перечисление диапазона имен BUILTIN при помощи PsGetSid

Следующий код PowerShell получает имена первых десяти локальных групп и пользователей, определенных на компьютере. Первая команда извлекает машинный SID из выходных данных PsGetSid, а вторая добавляет к этому идентификатору числа от 1000 до1009 и передает полученные значения утилите PsGetSid:

```
$msid = $(psgetsid)[2]  + «-»
1000..1009 | %{ psgetsid $msid$_ 2> $nul }
```

PsInfo

PsInfo собирает ключевую информацию о системах, включая тип установки, номер сборки ядра, время работы системы, имя и организацию зарегистрированного владельца, число процессоров и их тип, объем памяти и версию Internet Explorer. Параметры командной строки позволяют также просматривать информацию о дисках, установленных исправлениях и приложениях. Например:

```
System information for \\WIN7-X86-VM:
Uptime:                    0 days 23 hours  58 minutes 9 seconds
Kernel version:            Windows 7 Ultimate,  Multiprocessor Free
Product type:              Professional
Product version:           6.1
Service pack:              0
Kernel  build number:      7600
Registered organization:   Microsoft
Registered owner:          Abby
IE version:                8.0000
System  root:              C:\Windows
Processors:                1
Processor speed:           2.3 GHz
Processor type:            Intel(R)  Core(TM)2 Duo CPU   T7700  @
Physical   memory:         2048 MB
Video driver:              Microsoft Virtual  Machine Bus Video Device
```

Значение Uptime представляет суммарное время работы системы с момента последней загрузки. Время, проведенное в режиме ожидания и спящем режиме, не считается, так что это значение не обязательно показывает, сколько времени прошло с момента последней загрузки системы.

Примечание В момент написания книги объем физической памяти не сообщался для 64-разрядных версий Windows.

Чтобы получить только отдельные строки из этих данных, укажите в командной строке полное или частичное имя нужного поля или полей. Например, если выполнить команду psinfo register, будут показаны только поля Registered organization и Registered owner.

По умолчанию PsInfo получает информацию о локальном компьютере, но, используя синтаксис, описанный в начале главы, можно получать све-

дения об одном или нескольких удаленных компьютерах. PsInfo не требует прав администратора на локальном компьютере, но требует их на удаленных системах.

Если добавить параметр –d, будет показана также информация о томах дисков:

```
Volume    Type         Format     Label       Size         Free         Free
     A:   Removable                                                     0.0%
     C:   Fixed        NTFS                   126.99 GB    123.34 GB   97.1%
     D:   CD-ROM       CDFS       VMCUEST     23.66 MB                  0.0%
     X:   Remote       NTFS                   19.99 GB     13.35 GB    66.8%
```

В предыдущем примере у пользователя, запустившего PsInfo, был диск X:, назначенный удаленному хранилищу файлов. При опросе удаленных компьютеров PsInfo собирает информацию в контексте учетной записи системы, поэтому показываются только глобально видимые тома. Буквы, назначенные удаленным томам, входят сюда, только если они созданы в контексте учетной записи SYSTEM, что делает их видимыми всем процессам компьютера.

Примечание PsInfo не различает подстановки, сделанные с помощью SUBST. Если буква диска связана с локальным путем, она появится в списке как фиксированный диск с теми же характеристиками, что и у реального тома.

Если указать параметр –h, будут перечислены исправления, установленные на целевой системе. Сведения об исправлениях собираются из нескольких известных мест реестра, содержащих информацию об исправлениях Windows и Internet Explorer.

Параметр –s сообщает об установленных приложениях в соответствии с информацией для удаления приложений, хранящейся в реестре.

Можно получить выходные данные в виде значений, разделенных запятыми (CSV-файла), добавив параметр –c. Результаты с каждого компьютера выводятся в отдельной строке, что удобно для генерации электронной таблицы. Чтобы использовать другой символ-разделитель, добавьте параметр –t, за которым укажите новый разделитель. Чтобы использовать в качестве разделителя символ табуляции, укажите параметр \t:

```
psinfo -c -t \t
```

Если PsInfo выводит информацию о локальном компьютере или одном удаленном компьютере, кодом выхода PsInfo будет номер пакета обновлений, установленного в этой системе. При выводе информации о нескольких системах, PsInfo возвращает обычный код успешного завершения или сбоя.

PsKill

PsKill — утилита командной строки для завершения процессов, заданных по идентификатору или имени образа. Она также может завершать все произ-

водные процессы целевого процесса. Как и другие утилиты PsTools, PsKill может завершать процессы и деревья процессов на удаленных компьютерах, при необходимости используя альтернативные учетные данные.

Внимание! PsKill завершает процессы незамедлительно. Принудительное завершение процесса не дает ему возможности корректно завершить работу и может привести к потере данных или нестабильности системы. Кроме того, PsKill не показывает дополнительных предупреждений, если попытаться завершить процесс, критически важный для работы системы, такой как Csrss.exe. Завершение критически важного процесса приводит к появлению «синего экрана» Windows.

Чтобы завершить процесс, укажите в командной строке PsKill его идентификатор (PID) или имя образа. Если параметр можно интерпретировать как десятичное число, он считается идентификатором; в противном случае он считается именем образа. В имени образа не обязательно указывать расширение .exe, но в остальном оно должно быть точным — PsKill не принимает символы подстановки. Если указать имя образа, PsKill попытается завершить все процессы с таким именем.

Если имя образа является десятичным числом, включите в имя расширение .exe, чтобы этот параметр интерпретировался как имя, а не PID.

Если добавить параметр –t, будет завершено дерево процессов целевого процесса или процессов. Дерево процессов целевого процесса — это сам этот процесс и все созданные им процессы. Дерево процессов можно увидеть в Process Explorer (см. гл. 3) или при помощи параметра –t, о котором говорится далее в этой главе.

Для завершения процесса, работающего в том же контексте безопасности на том же компьютере, что и PsKill, администраторские права не требуются, но необходимы во всех остальных случаях.

Примечание Утилита PsKill была создана, когда в Windows было относительно немного утилит командной строки. В Windows XP появились утилиты Taskkill.exe и Tskill.exe, предоставляющие все возможности PsKill и даже больше.

PsList

PsList, первая из написанных мной утилит PsTools, основана на утилите *ps* с платформы UNIX. Она выводит список процессов и их характеристики, такие как использование памяти и процессора. Также PsList может показывать отношения родительских и дочерних процессов, выводить сведения для отдельных потоков и непрерывно обновляться в режиме диспетчера задач. Сведения можно получать для локальных и удаленных процессов.

Для отображения информации о процессах локальной системы администраторские права не требуются. По умолчанию для получения сведений об удаленном компьютере Windows XP требуются администраторские права на целевой системе. В Windows Vista и последующих версиях члены групп **Администраторы (Administrators)**, **Пользователи монитора производительности (Performance Monitor Users)** и **Пользователи журнала произ-**

водительности (Performance Log Users) могут выполнять PsList удаленно. На целевом компьютере должна быть запущена служба Remote Registry (Удаленный реестр).

При запуске без аргументов командной строки PsList выводит список процессов, работающих на локальном компьютере, в порядке их запуска; их идентификаторы (столбец с заголовком Pid); приоритет (Pri); число потоков (Thd); число описателей объектов ядра (Hnd); объем закрытой виртуальной памяти в килобайтах (Priv); общее время ЦП, занятое процессом; время с момента запуска процесса.

Примечание PsList использует имя «Idle» для ссылки на псевдо-процесс, имеющий PID 0, который в Process Explorer и других утилитах называется «System Idle Process» (Процесс бездействия системы). Как и большинство других утилит для просмотра процессов, PsList не идентифицирует отдельно псведо-процесс Interrupts (Прерывания), который идентифицируется в Process Explorer, вместо этого приписывая его нагрузку на ЦП процессу Idle.

Параметр –t отображает процессы в виде дерева аналогично Process Explorer, где дочерние процессы расположены под родительскими с отступом от края. В древовидном представлении столбцы **CPU Time (Время процессора)** и **Elapsed Time (Затраченное время)** не отображаются; вместо них PsList показывает зарезервированную виртуальную память (VM) и рабочий набор (WS) в килобайтах.

Параметр –m показывает для каждого процесса информацию, относящуюся к памяти, а не к процессору. Показанные сведения включают зарезервированную виртуальную память (VM), размер рабочего набора (WS), закрытую виртуальную память (Priv), пиковый объем закрытой виртуальной памяти за время жизни процесса (Priv Pk), страничные ошибки (Faults), включая жесткие и мягкие сбои, а также размеры неподкачиваемого и подкачиваемого пулов (NonP и Page, соответственно). Все размеры памяти показаны в килобайтах.

Параметр –d показывает информацию о каждом потоке в системе. Потоки группируются по процессам, которым принадлежат, и сортируются по времени запуска. Выводится идентификатор потока (Tid), приоритет потока (Pri), число переключений контекста или число раз, когда поток начинал выполнение на процессоре (Cswitch), текущее состояние потока (State), время выполнения в режиме пользователя (User Time) и в режиме ядра (Kernel Time), а также затраченное время (Elapsed Time) с момента начала выполнения потока.

Параметр –x показывает информацию о процессоре, памяти и потоках для каждого процесса. Параметры –m, -x и –d можно комбинировать, но нельзя использовать вместе с параметром –t.

Можно отображать информацию об отдельных процессах, указав их идентификаторы, частичные или точные имена. Следующая команда выводит информацию о процессе с PID 560 на компьютере с именем Win7_vm:

```
pslist \\Win7_vm 560
```

Следующая команда отображает информацию о процессоре, потоках и памяти для всех процессов, чье имя начинается с «svc»:

```
pslist -x svc
```

Параметр –e указывает, что введенное имя должно в точности соответствовать имени процесса. Если добавить этот параметр в предыдущий пример, будут показаны только процессы svc.exe, но не экземпляры svchost.exe.

Параметр –s запускает PsList в режиме «диспетчера задач», в котором экран консоли периодически очищается и обновляется последней информацией. Список сортируется по столбцу CPU, в котором показан процент времени ЦП, использованный каждым процессом с момента предыдущего обновления. По умолчанию PsList обновляет экран один раз в секунду до нажатия клавиши Escape. Можно указать длительность работы PsList, указав число секунд сразу после параметра –s, и изменить частоту обновления, добавив параметр –r. В следующем примере PsList будет работать в режиме диспетчера задач 60 секунд (или до нажатия Escape), обновляя экран каждые пять секунд:

```
pslist -s 60 -r 5
```

Параметр –s можно комбинировать с параметром –m, чтобы просматривать непрерывно обновляемые сведения о памяти и сортировать процессы по закрытым байтам, а не использованию ЦП. Можно также комбинировать –s и –t, чтобы просматривать обновляемые сведения о процессах в древовидном представлении, как в Process Explorer. Можно указать PID, частичное или точное имя процесса, чтобы просматривать информацию о них в режиме диспетчера задач. Если указывается PID, его лучше поставить перед параметром –s, чтобы он не интерпретировался как число секунд работы. Следующая команда позволяет непрерывно следить за использованием памяти процессом leakyapp.exe на удаленном компьютере:

```
pslist \\Win7_vm -s -m -e leakyapp
```

PsLoggedOn

PsLoggedOn сообщает, кто вошел в систему на определенном компьютере локально или удаленно. Можно также выяснить, на какие компьютеры в сети вошел определенный пользователь.

При запуске без параметров командной строки эта утилита сообщает, какие пользователи и когда вошли локально на текущий компьютер. Затем она сообщает, какие пользователи вошли через общие ресурсы, и в какое время были начаты сеансы. (Последняя возможность похожа на работу команды net session.)

Чтобы просмотреть эту информацию для удаленного компьютера, укажите его имя в командной строке:

```
psloggedon \\Win7_vm
```

Для запуска PsLoggedOn на удаленном компьютере требуется учетная запись с правами администратор. Это одна из утилит PsTools, не принимающих параметры –u и –p для указания альтернативных учетных данных. Поскольку PsLoggedOn использует для сбора информации на удаленном компьютере службу Remote Registry, она всегда будет показывать подключение к общему ресурсу на компьютере, с которого получается информация.

Чтобы отображались только локальные входы в систему, но не подключения к общим ресурсам, добавьте в командную строку параметр –l (L в нижнем регистре). Чтобы отображались только имена учетных записей, без времени входа, добавьте параметр –x.

Если указать имя пользователя, а не компьютера, PsLoggedOn выполнит поиск по всем компьютерам текущего домена или рабочей группы и сообщит, на каких системах этот пользователь локально вошел в систему. Обратите внимание: учетная запись, от имени которой запущена утилита, должна иметь администраторские права на всех компьютерах сети, а поиск может занимать много времени в крупных сетях и сетях с ограниченной пропускной способностью.

PsLoggedOn считает, что пользователь вошел в систему локально, если его профиль загружен в реестр. Когда профиль загружается, SID пользователя появляется в разделе HKEY_USERS в качестве подраздела. PsLoggedOn проверяет последнюю временную отметку подраздела с этим SID в качестве имени, чтобы определить примерное время входа пользователя в систему. Определенное время будет точным в большинстве случаев. Для получения более полной и точной информации о сеансах используйте утилиту LogonSessions, описанную в главе 8.

PsLogList

PsLogList отображает записи из журналов событий локального или удаленных компьютеров. Можно фильтровать выходные данные по временным отметкам, источнику, идентификатору и другим условиям. Также эта утилита позволяет экспортировать записи журнала в evt-файл, прочитать записи из сохраненного evt-файла и очистить журнал событий.

При запуске без параметров PsLogList выгружает все записи из журнала событий Система (System) локального компьютера. Чтобы просмотреть записи из другого журнала событий, укажите его имя в командной строке. Например, следующие команды выгружают записи из журнала Приложения (Application) и журнала PowerShell, соответственно:

```
psloglist application
psloglist “Windows Powershell”
```

Чтобы просмотреть записи с одного или нескольких удаленных компьютеров, укажите имена систем в командной строке, как описано в начале главы.

Каждая запись журнала событий включает источник события и идентификатор события. Идентификатор события используется для поиска и ото-

бражения локализуемого, понятного текста из библиотеки DLL с ресурсами сообщений, связанной с источником события. Текст сообщения может содержать заполнители для текста, изменяющегося для разных событий (такого как имя файла и IP-адрес). Этот текст связан с записью журнала событий в виде *подстановочных строк вставки*, которых может быть ноль или более. Большинство приложений для просмотра событий, включая стандартную утилиту Windows **Просмотр событий (Event Viewer)**, отображают только подстановочные строки, а не полный текст, когда встречаются ссылки на библиотеки DLL с ресурсами сообщений, которых нет на локальной системе. Это делает текст сложным для чтения. Одна из функций, выгодно отличающих PsLogList при чтении удаленных журналов событий, — получение текста сообщений из DLL ресурсов, расположенных на этих удаленных системах. Однако для этого на удаленной системе должна быть включена и доступна стандартная административная общая папка Admin$, библиотеки DLL ресурсов должны находиться в этой папке, а служба Remote Registry запущена. Прежде чем собирать данные с удаленных систем, убедитесь, что эти условия выполняются, иначе PsLogList не сможет отображать полный текст событий.

PsLogList не требует прав администратора для отображения записей из локального журнала приложений, системного журнала или сохраненных evt -файлов, а также для экспорта записей из этих журналов в файл формата evt. Администраторские права могут не потребоваться для просмотра журнала приложений удаленной системы Windows XP, но текст событий будет недоступен. Администраторские права требуются для очистки журналов событий и для доступа к локальному журналу безопасности (Security) и любым другим удаленным журналам событий.

Остальные параметры командной строки PsLogList вкратце описаны в табл. 6-2 и подробнее обсуждаются ниже.

Табл. 6-2. Параметры командной строки PsLogList

Параметр	Описание
Параметры вывода	
-x	Отображает расширенные данные, если таковые имеются. (Неприменим, если используется параметр –s)
-n #	Ограничивает число отображаемых записей до указанного номера
-r	Обращает порядок сортировки: первыми отображаются самые старые записи, потом более новые (по умолчанию первыми отображаются новые записи)
-s	Отображает каждую запись в одной строке с разделенными полями
-t *символ*	Указывает символ-разделитель, используемый с параметром –s. Чтобы использовать символ табуляции, укажите \t
-w	Ожидает новые события, отображая их при возникновении. PsLogList работает до завершения нажатием Ctrl+C (только на локальном компьютере)

Табл. 6-2. (*окончание*)

Параметр	Описание
Параметры временных отметок	
-a *мм/дд/гггг*	Отображает записи с временными штампами не раньше указанной даты
-b *мм/дд/гггг*	Отображает записи с временными штампами до указанной даты
-d #	Отображает только записи за предыдущие # дней
-h #	Отображает только записи за предыдущие # часов
-m #	Отображает только записи за предыдущие # минут
Параметры фильтрации содержимого событий	
-f *фильтр*	Фильтрует типы событий, де каждая буква в значении *фильтр* представляет тип события
-i *ID* [,*ID*,...]	Отображает только события с указанными ID (до 10 штук)
-e *ID* [,*ID*,...]	Отображает события *за исключением* событий с указанными ID (до 10 штук)
-o *источник,* [*источник*,...]	Отображает только события из указанных источников событий. Можно применять символ * для сопоставления подстрок
-q *источник,* [*источник*,...]	Отображает события *за исключением* событий из указанных источников. Можно применять символ * для сопоставления подстрок
Параметры управления журналом	
-z	Отображает список журналов событий, зарегистрированных на целевой системе
-c	Очищает журнал событий после отображения записей
-g *имя_файла*	Экспортирует журнал событий в файл *.evt (только на локальном компьютере)
-l *имя_файла*	Отображает записи из сохраненного файла *.evt, а не из активного журнала

По умолчанию PsLogList отображает для каждой записи номер, источник, тип, временную отметку, идентификатор события и текст описания. PsLogList загружает исходные модули сообщения на систему, где находится просматриваемый журнал событий, чтобы она корректно отображала сообщения из журнала событий. Например:

```
[34769]   Service Control  Manager
  Type:        INFORMATION
  Computer:    WIN7X86-VM
  Time:        12/22/2009 11:31:09   ID:       7036
The Application  Experience service entered the stopped state.
```

Параметр `-x` отображает любые расширенные данные из журнала событий в шестнадцатеричном формате. С этим параметром предыдущий пример будет выглядеть так:

```
[34769]   Service Control Manager
  Type:        INFORMATION
  Computer:    WIN7X86-VM
  Time:        12/22/2009 11:31:09   ID:       7036
The Application  Experience service entered the stopped state.
  Data:
  0000: 41 00 65 00 4C 00 6F 00 6F 00 6B 00 75 00 70 00 A.e.L.o.o.k.u.p.
  0010: 53 00 76 00 63 00 2F 00 31 00 00 00             S.v.c./.1...
```

Параметр –n ограничивает число отображаемых записей указанным числом. Следующая команда отобразит 10 последних записей из журнала приложений:

```
psloglist -n 10 application
```

По умолчанию PsLogList отображает записи в порядке от новых к старым. Параметр `-r` обращает этот порядок, и сначала отображаются старые записи. В следующей команде скомбинированы параметры `-r` и `-n`, чтобы отобразить 10 самых старых записей из журнала приложений:

```
psloglist -r -n 10 application
```

Параметр –s отображает содержимое каждой записи в отдельной строке с полями, разделенными запятыми. Это удобно для поиска текста, так как можно искать в записи любой текст и видеть запись целиком. Например, psloglist –s | findstr /I luafv. Параметр –t позволяет указать другой символ-разделитель, что облегчает импортирование в электронную таблицу. Заметьте, что PsLogList в режиме –s заключает в кавычки только текст описания, так что используйте разделитель, не встречающийся в тексте событий. Можно использовать параметр \t, чтобы выбрать в качестве разделителя знак табуляции. Также обратите внимание на то, что расширенные данные (-x) не выводятся при использовании параметра –s.

Если указать параметр –w, PsLogList запустится в непрерывном режиме, ожидая и отображая новые события при их добавлении в журнал событий. Можно комбинировать этот параметр с другими параметрами фильтрации, чтобы отображать только новые записи, удовлетворяющие указанным условиям. PsLogList будет продолжать работу до нажатия Ctrl+C или Ctrl+Break. Параметр –w нельзя использовать при получении сведений с удаленного компьютера.

Параметры –a и –b фильтруют записи на основе их временных штампов. Параметр –a отображает только записи, добавленные не раньше указанной даты; параметр –b отображает только записи, добавленные после указанной даты. Дата указывается в формате месяц/день/год, независимо от текущих региональных параметров форматирования дат. Следующая команда отображает все записи из журнала Система (System) за 22 декабря 2009 г.:

```
psloglist -a 12/22/2009 -b 12/23/2009
```

Вместо того, чтобы указывать определенную дату, можно получить последние записи из журнала событий за определенный период времени. Параметры –d, -h и –m позволяют показать наиболее свежие записи за указанное число дней, часов или минут, соответственно. Следующая команда отображает все записи из журнала Система (System), произошедшие за последние три часа:

```
psloglist -h 3
```

Параметр –f фильтр фильтрует записи по типу события. Для каждого типа событий, который должен отображаться, нужно добавить соответствующую букву. Например, -f e отображает только события ошибок, -f ew — ошибки и предупреждения, а –f f — сбои аудита. Для информационных событий указывается буква i, а для успешного аудита — s.

Чтобы вывести только записи с определенными идентификаторами событий, используйте параметр –i, за которым можно указать до 10 идентификаторов, разделенных запятыми. Чтобы исключить события с определенными идентификаторами, используйте параметр –e. Список не должен содержать пробелов.

Чтобы просмотреть события из определенных источников событий, используйте параметр –o, за которым укажите список имен источников, разделенный запятыми. Если имя какого-либо источника содержит пробелы, заключите в кавычки весь список. Можно использовать символ *, чтобы заменять любой текст в имени источника. Не вставляйте пробелы рядом с запятыми. Чтобы не включить, а исключить записи из определенных источников, используйте параметр –q вместо –o. В следующем примере отображаются все события из журнала Система (System) с источником Service Control Manager или любым источником, в имени которого имеется текст «net», но исключаются записи с идентификаторами 1 и 7036.

```
psloglist -o «service control manager,net*» -e 1,7036
```

Можно экспортировать журнал событий локального компьютера в evt -файл, используя параметр –g. Следующая команда экспортирует журнал Приложение (Application) в файл **app.evt**, сохраняемый в текущей папке:

```
psloglist -g .\app.evt Application
```

Можно просматривать записи из сохраненного evt-файла, а не из активного журнала событий, используя параметр –l (L в нижнем регистре). Чтобы текст событий интерпретировался корректно, укажите также исходное имя журнала. Следующая команда отображает 10 последних записей из сохраненного файла app.evt, используя файлы сообщений, связанные с журналом Приложение (Application):

```
psloglist -l .\app.evt -n 10 application
```

PsLogList поддерживает просмотр только журналов событий в старом формате. Параметр –z позволяет просмотреть список журналов событий,

доступных для просмотра на целевой системе. Заметьте, что зарегистрированное имя журнала событий может отличаться от имени, отображаемого утилитой Просмотр событий (Event Viewer).

Наконец, можно очистить журнал событий после отображения записей, указав параметр –c. Чтобы не отображать записи, используйте фильтр, исключающий все, например, -f x (ни один тип события не начинается с «x»). Следующая команда очищает журнал Безопасность (Security) на удаленном компьютере, не отображая записи из него:

```
psloglist \\win7demo -c -f x security
```

PsPasswd

PsPasswd позволяет устанавливать пароли для локальных или доменных учетных записей пользователей. Можно установить пароль для именованной локальной учетной записи на одном компьютере, наборе компьютеров или всех компьютеров домена или рабочей группы. Это может быть полезным для установки паролей для учетных записей служб или локальных встроенных учетных записей администратора.

Чтобы установить пароль в домене, просто укажите целевую учетную запись в формате домен\учетная запись и новый пароль. Если имя учетной записи или пароль содержат пробелы, заключайте их в кавычки. В следующем примере устанавливается очень сложный, но легко запоминаемый пароль длиной 28 символов для учетной записи MYDOMAIN\Toby:

```
pspasswd mydomain\toby «Passphrase++ 99.9% more good»
```

Пароль указывать не обязательно. Если указать только имя учетной записи, PsPasswd установит пустой пароль, если это разрешено политикой безопасности.

Чтобы установить пароль для учетной записи на локальном компьютере, укажите только имя учетной записи и пароль. Пароль так же не обязателен; если его не указать, будет установлен пустой пароль, если это разрешено политикой безопасности.

Примечание Сброс пароля локальной учетной записи пользователя может привести к безвозвратной потере зашифрованных данных, принадлежащих этой учетной записи, таких как файлы, защищенные шифрующей файловой системой (EFS).

Чтобы установить пароль для локальной учетной записи на одном или нескольких удаленных компьютерах, используйте синтаксис, описанный в начале главы, при необходимости указав альтернативные учетные данные. Затем укажите имя учетной записи и новый пароль (его указывать необязательно). В следующем примере локальной учетной записи Administrator на всех компьютерах домена или рабочей станции назначается случайный пароль из 50 символов:

```
pspasswd \\* Administrator «^HVKh*iK:F`Rv0[8<Zdp#|,I.:TI_K':W\xEwi9D3I,0}tQ>tK»
```

По умолчанию устанавливать пароли учетным записям домена могут только администраторы домена (Domain Admins) и операторы учетных записей (Account Operators). Заметьте, что PsPasswd не принимает альтернативные учетные данные в случае с учетными записями домена; эту утилиту следует запускать с привилегиями, необходимыми для изменения целевого пароля. Чтобы установить пароль для локальной учетной записи, требуются администраторские права на целевом компьютере.

PsService

PsService отображает список и управляет службами Windows и драйверами на локальной или удаленной системе. Эта утилита во многом аналогична SC.EXE и некоторым функциям NET.EXE (обе эти утилиты входят в состав Windows), но является более гибкой и удобной в использовании. Например, службы можно указывать по имени, отображать имена, а в некоторых случаях выполнять сопоставление по частичному имени. Кроме того, PsService включает уникальную возможность поиска служб, позволяющую искать экземпляры службы в сети, а также службы, помеченные как «интерактивные».

При запуске без параметров PsService выводит информацию о состоянии всех служб Win32 (работающих в пользовательском режиме), зарегистрированных на локальном компьютере. Конечно, можно указать в командной строке имя удаленного компьютера и предоставить альтернативные учетные данные, если текущая учетная запись не имеет прав администратора на удаленной системе.

PsService поддерживает следующие команды и параметры, которые будут подробно обсуждаться в этом разделе:

- query [-g group] [-t {driver|service|interactive|all}] [-s {active|inactive|all}] [service]
- config [service]
- depend service
- security service
- find service [all]
- setconfig service {auto|demand|disabled}
- start service
- stop service
- restart service
- pause service
- cont service

Получить список этих параметров можно командой PsService /?. Команда PsService команда /? отображает синтаксис указанной команды, например, psservice query /?.

PsService явно не требует прав администратора для работы на локальном компьютере. Поскольку разрешения для каждой службы можно устанавли-

вать по отдельности, разрешения, необходимые для локальной работы, могут различаться в зависимости от того, какие службы задействованы. Например, некоторые (на самом деле, немногие) службы предоставляют интерактивному пользователю разрешение на запуск и остановку этой службы. Другой пример: команда psservice depend server показывает список служб, зависимых от службы Server. Полученный список в Windows 7 будет различаться для администраторов и не администраторов, так как последним не разрешено считывать информацию о состоянии службы HomeGroup Listener, которая зависит от службы Server.

Query

Команда query отображает информацию о состоянии служб и драйверов целевой системы, используя гибкие критерии фильтрации. Для каждой подходящей службы или драйвера PsService показывает следующие сведения:

- имя службы — внутреннее имя службы или драйвера. Это имя, которое требует большинство команд **sc.exe**;
- отображаемое имя — имя, отображаемое в оснастке MMC **Службы (Services)**;
- описание — текст описания, связанный со службой или драйвером;
- группа — группа порядка загрузки, в которую входит служба, если указана;
- тип — службы, работающие в пользовательском режиме, являются либо службами с собственным процессом, либо службами с общим процессом, в зависимости от того, может ли процесс службы размещать другие службы. Процессы пользовательского режима могут также быть помечены как «интерактивные» (хотя это очень не рекомендуется). Драйверы могут быть драйверами ядра или драйверами файловой системы. (Драйверы файловой системы должны регистрироваться диспетчером ввода-вывода; они более активно взаимодействуют с диспетчером памяти.);
- состояние — указывает, работает служба, остановлена, приостановлена или переходит к запуску, остановке, приостановке или продолжению работы. Под этой строкой PsService показывает, принимает ли служба команды приостановки и продолжения работы и может ли обрабатывать уведомления, отправляемые перед выключением и при выключении;
- код завершения Win32 — ноль указывает на нормальную работу и завершение. Ненулевое значение указывает на стандартный код ошибки, сообщенный службой. Значение 1066 указывает ошибку, специфичную для службы. Значение 1077 указывает, что служба не была запущена после последней загрузки, что нормально для многих служб;
- код завершения, специфичный для службы — если Код завершения Win32 равен 1066 (0x42A), это значение указывает на код ошибки, специфичный для службы; в противном случае оно не имеет смысла;

- контрольная точка — это значение, обычно равное нулю, периодически увеличивается, чтобы сообщить о выполнении службой затянувшихся операций по запуску, остановке, приостановке и продолжению работы. Оно не имеет смысла, когда операция не выполняется;
- подсказка ожидания — время в миллисекундах, которое, как предполагает система, потребуется для выполнения предстоящей операции запуска, остановки, приостановки или продолжения работы. Если это время истекает без изменения состояния или контрольной точки, можно считать, что в службе произошла ошибка.

По умолчанию команда PsService query выводит список всех служб Win32, настроенных на целевой системе, независимо от того, запущены они или нет. (Запуск PsService без параметров командной строки аналогичен выполнению команды psservice query.) Чтобы отобразить информацию об определенной службе или драйвере, укажите в конце командной строки соответствующее имя. PsService сообщит информацию о состоянии всех служб и драйверов, имена служб или отображаемые имена которых частично или полностью соответствуют указанному имени. Например, команда psservice query ras выведет все службы и драйвера, имена которых начинаются с «ras» (сопоставление выполняется без учета регистра).

Можно продолжить фильтрацию результатов по типу и состоянию. Добавьте параметр –t и укажите driver, чтобы отображать только драйверы, service, чтобы отображать только службы Win32, interactive, чтобы отображать только службы Win32, для которых помечен флажок **allow service to interact with desktop** (разрешить службе взаимодействие с рабочим столом), или all, чтобы не фильтровать результаты по типу. Чтобы фильтровать результаты по активности служб и драйверов, добавьте параметр –s и укажите active, inactive или all. Если в командной строке не указано имя службы, PsService по умолчанию будет отображать только службы Win32 и все состояния. Если имя службы указано, а параметр –t нет, будут отображаться службы и драйверы, соответствующие выбранным условиям.

Примечание Настоятельно рекомендуется не помечать службы как «интерактивные». Такие службы уязвимы для атак с повышением привилегий и часто не работают в Windows Vista и выше, а также в ранних версиях Windows с быстрым переключением пользователей или другими службами терминалов. Команда psservice query –t interactive — простой способ выявления таких потенциально проблемных служб.

Чтобы получить список только тех служб и драйверов, которые входят в определенную группу порядка загрузки, укажите имя группы после параметра –g. Сопоставление имени группы выполняется без учета регистра, но совпадение должно быть полным, а не частичным.

Все эти параметры можно комбинировать. Приведенная ниже команда отображает информацию о состоянии драйверов ядра на удаленном компьютере, входящих в группу **PnP Filter**, которая не загружается, имеющих отображаемые имена или имена служб, начинающиеся с «bth».

```
psservice \\win7x86-vm query -g «pnp filter» -t driver -s inactive bth
```

Config

Команда config отображает сведения о конфигурации служб и драйверов. При запуске без параметров эта команда показывает сведения о конфигурации всех служб Win32, зарегистрированных на целевой системе. Если добавить после команды config имя службы, PsService покажет все параметры конфигурации всех служб и драйверов, чьи имена начинаются с указанного текста. Например, команда psservice config ras отображает параметры конфигурации всех служб и драйверов, чьи имена служб или отображаемые имена начинаются с «ras» (регистр не учитывается).

Команда config отображает следующую информацию:

- имя службы — внутреннее имя службы или драйвера. Это имя указывается в большинстве команд **sc.exe**;
- отображаемое имя — имя, отображаемое в оснастке MMC Службы (Services);
- описание — текст описания, связанный со службой или драйвером;
- тип — указывает, как настроен элемент: как служба с собственным процессом или как служба с общим процессом и помечен ли как «интерактивный»; а если это драйвер, то настроен ли он как драйвер ядра или как драйвер файловой системы;
- тип запуска — драйверы, загружающиеся при загрузке системы, могут быть помечены как запускаемые при загрузке или запускаемые системой; службы, загружающиеся при загрузке системы, могут быть помечены как запускаемые автоматически или запускаемые автоматически с задержкой. Пометка «demand-start» (запуск по запросу) или «manual start» (ручной запуск) указывает на то, что службы и драйверы могут запускаться при необходимости. Службы и драйверы с пометкой «Disabled» (отключен) не могут быть загружены;
- контроль ошибок — указывает, что следует предпринять Windows, если службе или драйверу не удалось запуститься во время загрузки системы. Значения **Ignore** и **Normal** означают, что Windows будет продолжать загрузку системы, записав ошибку в журнал событий при значении **Normal**. При значениях **Severe** и **Critical** Windows выполняет перезапуск с использованием последней корректной конфигурации. Если ошибка происходит и в этом случае, то загрузка продолжается при значении **Severe**, но прекращается при значении **Critical**;
- двоичное имя пути — показывает путь к исполняемому файлу, который должен быть загружен, и необязательные параметры командной строки для автоматически запускаемых служб;
- группа порядка загрузки — имя группы порядка загрузки, в которую входит служба или драйвер (пустое значение, если такой группы нет);
- тег — для драйверов, запускаемых при загрузке и запускаемых системой, являющихся частью группы порядка загрузки, это уникальное значение

в группе, которое можно использовать для указания порядка загрузки в группе;

- зависимости — службы или группы порядка загрузки, которые должны быть загружены, чтобы служба или драйвер могли запуститься;
- имя запуска службы — для служб это имя учетной записи, от имени которой служба запускается.

Depend

Команда depend показывает список служб и драйверов, имеющих прямые или косвенные зависимости от указанной службы. Например, команда psservice depend tdx показывает службы и драйверы, которые не могут быть запущены, если не загружен драйвер tdx (NetIO Legacy TDI Support Driver).

Команда depend показывает ту же информацию, что и команда query. Имя службы в командной строке psservice depend должно точно соответствовать имени службы или отображаемому имени зарегистрированной службы или драйвера. Для этой команды PsService не выполняет сопоставление по частичному имени.

Чтобы посмотреть, от каких служб зависит определенная служба, используйте команду psservice config.

Security

Как можно догадаться, команда security отображает информацию о безопасности указанной службы или драйвера. В частности, она показывает список избирательного контроля доступа (DACL) в удобочитаемом виде. Вместо того, чтобы использовать малопонятный язык определения дескрипторов безопасности (Security Descriptor Definition Language, SDDL), она выводит имена учетных записей, которым предоставлен или запрещен доступ, и соответствующие им разрешения.

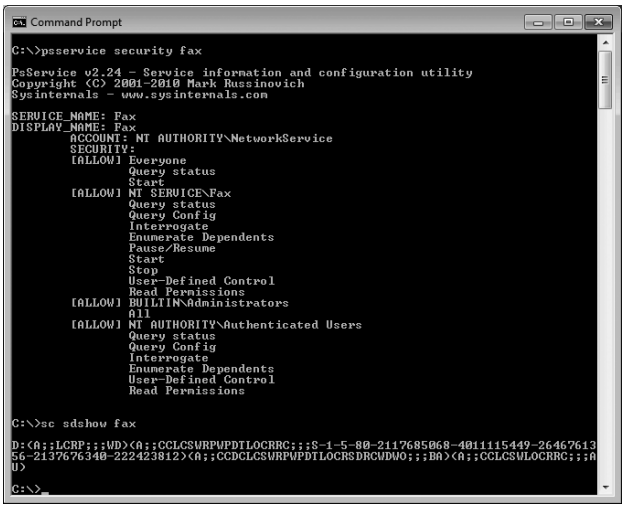


Рис. 6-2. Команда PsService security и эквивалентная команда SC.EXE

Из рис. 6-2 видно, что, по данным PsService, службу Fax могут запускать все пользователи. Эквивалентные, но мене понятные сведения на языке SDDL показывает утилита sc.exe (см. на том же рисунке).

Для служб Win32 PsService также показывает имя учетной записи, от имени которой запущена служба.

В командной строке PsService security необходимо указывать точное имя службы или отображаемое имя службы или драйвера, без учета регистра.

Find

Одна из уникальных возможностей PsService — поиск в сети экземпляров службы. Команда find перечисляет все компьютеры рабочей группы или домена и проверяет каждый из них на наличие запущенного экземпляра указанной службы. Поиск можно выполнять по имени службы или отображаемому имени. Например, следующая команда показывает все компьютеры Windows в текущем домене или рабочей группе, на которых запущена служба DNS Server:

```
psservice find  «dns server"
```

Чтобы найти и запущенные, и неактивные экземпляры службы, добавьте ключевое слово all:

```
psservice find  «dns server»  all
```

Команду find можно использовать также для поиска в сети загруженных или неактивных драйверов. Например, команда psservice find vmbus позволяет найти компьютеры с загруженным драйвером Virtual Machine Bus.

SetConfig

Команда setconfig позволяет задать тип запуска службы Win32. Укажите имя службы или отображаемое имя и тип запуска. Доступны следующие параметры: auto для автоматического запуска службы, demand для запуска вручную и disabled для предотвращения запуска службы. Например, чтобы отключить службу Fax, выполните следующую команду:

```
psservice setconfig fax disabled
```

Start, Stop, Restart, Pause, Continue

При помощи PsService можно запускать, останавливать, перезапускать, приостанавливать и возобновлять работу службы. Синтаксис простой: введите команду start, stop, restart, pause или cont и имя или отображаемое имя службы или драйвера. Если управляющая команда выполнена успешно, PsService покажет результаты команды query, которая покажет, что запрошенная операция ожидается или выполнена. Заметьте, что не все эти операции допустимы для некоторых служб и драйверов. Также обратите внимание на то, что команды stop и restart не работают, если есть запущенные службы или драйверы, зависимые от указанной службы или драйвера.

PsShutdown

Утилита PsShutdown аналогична консольной утилите Shutdown.exe из старых версий Windows Resource Kit и текущих версий Windows. Она позволяет выключать, перезагружать или отправлять в спящий режим удаленные системы Windows из командной строки. В PsShutdown впервые появилась возможность указать причину выключения, которая затем была добавлена в Shutdown.exe.

Поскольку PsShutdown была создана до появления служб терминалов и распространения концепции учетных записей без прав администратора, она будет полезной в Windows до версии XP. PsShutdown требует прав администратора для создания и запуска собственной службы, которая выполняет большинство задач, а при выполнении пользовательских операций, таких как блокирование рабочей станции и выход из системы, предполагается, что службы и интерактивный рабочий стол пользователя находятся в том же сеансе служб терминалов («сеанс 0»). В Windows Vista и выше это не так, да и в Windows XP при использовании быстрого переключения пользователей и в Windows Server 2003 при использовании удаленного рабочего стола тоже. Однако PsShutdown имеет функцию, которой нет в Shutdown.exe — отправление компьютера в режим ожидания.

В табл. 6-3 описаны параметры командной строки PsShutdown. Чтобы предотвратить случайное использование, PsShutdown требует указания параметров командной строки.

Табл. 6-3. Параметры командной строки PsShutdown

Параметр	Описание
Команды выключения (требуется одна)	
-s	Выключает компьютер (питание остается включенным, если BIOS не поддерживает его отключение)
-k	Отключает питание компьютера (выполняет перезагрузку, если BIOS не поддерживает отключение питания)
-r	Перезагружает компьютер
-h	Отправляет компьютер в спящий режим
-d	Отправляет компьютер в режим ожидания
-l	Блокирует рабочую станцию (только в Windows XP и Windows 2003). Выполняется блокирование рабочей станции или отключение пользователя удаленного рабочего стола, если интерактивный пользователь работает в сеансе 0 служб терминалов. В противном случае не имеет эффекта
-o	Выполняет выход из системы (только в Windows XP и Windows 2003). Выводит из системы, если интерактивный пользователь работает в сеансе 0 служб терминалов. Если не указать параметр –f, выход из системы может быть блокирован приложением, отказавшимся завершить работу. В противном случае не имеет эффекта

Табл. 6-3. (*окончание*)

Параметр	Описание
-a	Отменяет операцию выключения компьютера, инициированную PsShutdown (действует, только когда выполняется такая операция). Эта команда не требует прав администратора при выпол-нении на текущем компьютере
Параметры отображения	
-m *«сообщение»*	Для операций выключения показывает интерактивному пользователю диалоговое окно с введенным сообщением. Если этот параметр не указан, показывает стандартное уведомление
-c	Для операций выключения добавляет в окно уведомления кнопку Cancel (Отмена), позволяющую отменить операцию
-v *секунды*	Перед выключением отображает окно уведомления в течение указанного времени. Если этот параметр не указан, диалоговое окно появляется сразу при инициировании команды на выключение. Если указать значение 0, диалоговое окно не будет показано
Другие параметры	
-t [*секунды*|*чч:мм*]	Указывает время выполнения операции выключения: либо задержку в секундах, либо время в 24-часовом формате. По умолчанию используется задержка в 20 секунд. (Этот параметр нельзя использовать с параметрами –l, -o и –a)
-f	Принудительно завершает работу приложений. (Заметьте, что Shutdown.exe в Windows XP/Windows 2003 имеет ошибку, из-за которой параметр –f действует «наоборот»)
-e [u|p]:*xx:yy*	Указывает код причины выключения: u — незапланированное, p — запланированное
-n *секунды*	Указывает таймаут в секундах для подключения к удаленным компьютерам

PsShutdown не использует API-функции *lnitiateSystemShutdown[Ex]* и *AbortSystemShutdown* для удаленного выключения или отмены выключения интерактивным пользователем. Вместо этого ее служба отображает собственное интерактивное диалоговое окно. Поэтому PsShutdown не может отменять операции, инициированные другими утилитами, и наоборот.

Диалоговое окно уведомления и отмены отображается службой PsShutdown, которая удаленно создается и настраивается как *интерактивная служба*. Интерактивные службы не рекомендуется использовать в Windows, поэтому диалоговое окно работает, как должно, только в некоторых случаях:

- в Windows XP и Server 2003 диалоговое окно отображается только интерактивному пользователю, работающему в сеансе 0 служб терминалов, и только если значение NoInteractiveServices не установлено. Если используется быстрое переключение пользователей или удаленный рабочий стол, пользователи могут работать в других сеансах. Пользователь сеанса 0 может отключиться или даже выйти из системы;

- в Windows Vista и последующих версиях, когда служба PsShutdown показывает уведомление, пользователь, интерактивно вошедший в систему, уведомляется службой обнаружения интерактивных служб — Interactive Services Detection (UI0Detect). Эта служба, если она не отключена, позволяет пользователям временно переключаться в сеанс 0 для взаимодействия с диалоговым окном. Если эта служба отключена, интерактивные пользователи не будут получать уведомления.

Причина, по которой может потребоваться использование параметра –n для управления тайм-аутом удаленного подключения, в следующем. Если попытаться использовать PsShutdown для управления компьютером, который уже выключен, команда может «зависнуть» на минуту до истечения тайм-аута. Эта задержка, являющаяся стандартным тайм-аутом Windows для подключения к другим компьютерам, может значительно увеличить время выполнения операций по выключению многих компьютеров одновременно. Параметр –n позволяет PsShutdown сократить время ожидания установки подключения, прежде чем отказаться от выполнения операции.

Ниже перечислены коды причин выключения, которые можно указывать с параметром –e:

Тип	Основное значение	Вспомогательное значение	Причина
U	0	0	Other (Unplanned) [Другая (незапланированная)]
P	0	0	Other (Planned) [Другая (запланированная)]
U	1	1	Hardware: Maintenance (Unplanned) [Оборудование: обслуживание (незапланированное)]
P	1	1	Hardware: Maintenance (Planned) [Оборудование: обслуживание (запланированное)]
U	1	2	Hardware: Installation (Unplanned) [Оборудование: установка (незапланированная)]
P	1	2	Hardware: Installation (Planned) [Оборудование: установка (запланированная)]
U	2	2	Operating System: Recovery (Unplanned) [Операционная система: восстановление (незапланированное)]
P	2	2	Operating System: Recovery (Planned) [Операционная система: восстановление (запланированное)]
P	2	3	Operating System: Upgrade (Planned) [Операционная система: обновление (запланированное)]
U	2	4	Operating System: Reconfiguration (Unplanned) [Операционная система: изменение конфигурации (незапланированное)

Тип	Основное значение	Вспомогательное значение	Причина
P	2	4	Operating System: Reconfiguration (Planned) [Операционная система: изменение конфигурации (запланированное)]
P	2	16	Operating System: Service pack (Planned) [Операционная система: установка пакета обновлений (запланированная)]
U	2	17	Operating System: Hot fix (Unplanned) [Операционная система: установка исправления (незапланированная)]
P	2	17	Operating System: Hot fix (Planned) [Операционная система: установка исправления (запланированная)]
U	2	18	Operating System: Security fix (Unplanned) [Операционная система: исправление системы безопасности (незапланированное)]
P	2	18	Operating System: Security fix (Planned) [Операционная система: исправление системы безопасности (запланированное)]
U	4	1	Application: Maintenance (Unplanned) [Приложение: обслуживание (незапланированное)]
P	4	1	Application: Maintenance (Planned) [Приложение: обслуживание (запланированное)]
U	4	5	Application: Unresponsive [Приложение: не отвечает]
U	4	6	Application: Unstable [Приложение: нестабильность]
U	5	19	Security issue [Вопросы безопасности]
P	5	19	Security issue [Вопросы безопасности]
U	5	20	Loss of network connectivity (Unplanned) [Разрыв сетевого подключения (незапланированный)]
P	7	0	Legacy API shutdown [Выключение с использованием устаревших API]

Журнал событий Система (System) может содержать ошибки, относящиеся к PsShutdown. Об отмене операции отключения может сообщаться как о неожиданном завершении службы PsShutdown; в журнале может также сообщаться об ошибках из-за того, что служба PsShutdown настроена как интерактивная служба. Ошибки обоих типов можно игнорировать.

PsSuspend

PsSuspend позволяет приостанавливать процессы на локальной или удаленной системе. Это может быть полезно, если процесс потребляет ресурсы (например, процессорное время), которые нужно предоставить другому процессу. Чтобы не завершать такой процесс, можно приостановить его, а позже возобновить его работу. Это также удобно при поиске и удалении вредоносного ПО, у которого несколько процессов, следящих за завершением друг друга.

Чтобы приостановить один процесс, укажите его идентификатор (PID) в командной строке PsSuspend. Можно указать имя процесса, чтобы приостановить все процессы с таким именем. Чтобы возобновить работу процесса, используйте параметр –r.

Каждый поток в процессе имеет счетчик пауз, так что каждый вызов API SuspendThread должен быть сопоставлен с вызовом ResumeThread, прежде чем поток сможет возобновить работу. PsSuspend сохраняет счет пауз потоков процесса. Это нужно, чтобы потоки, которые уже были в паузе, когда процесс приостанавливается при помощи PsSuspend, остались в паузе, когда работа процесса будет возобновлена. Если запустить PsSuspend с параметром –r для процесса, который не приостановлен, но имеет приостановленные потоки, счет приостановок этих потоков будет уменьшен, и если счет станет равным нулю, потоки возобновят работу. Программы, приостанавливающие свои потоки, скорее всего, имеют на то причины, так что следует внимательно относиться к «возобновлению» процессов, которые вы не приостанавливали.

Синтаксис командной строки PsTools

В этом разделе приведен синтаксис командной строки всех утилит PsTools. Так как синтаксис удаленного выполнения операций одинаков для всех утилит, покажем его отдельно, а не для каждой утилиты. Синтаксис указания нескольких компьютеров (УдаленныеКомпьютеры) применим ко всем утилитам, которые могут выполняться на нескольких компьютерах; синтаксис вида УдаленныйКомпьютер применим к утилитам, которые могут выполняться только на одном удаленном компьютере.

УдаленныеКомпьютеры = *компьютер*[,*компьютер2*[,...]] | * | @*файл* [-u *имя_пользователя* [-p *пароль*]]

УдаленныйКомпьютер = *компьютер* [-u *имя_пользователя* [-p *пароль*]]

PsExec

psexec [*УдаленныеКомпьютеры*] [-d] [-background|-low|-belownormal|-abovenormal|-high|-realtime] [-a n[,n[,...]]] [-c [-f|-v]] [-n *секунды*]

```
[-s|-e] [-i [сеанс]] [-x] [-w каталог] [-h] [-l] [-u имя_пользователя [-p
пароль]] команда [аргументы]
```

В отличие от других утилиты, PsExec поддерживает использование параметров –u и –p для удаленного и локального выполнения операций.

PsFile

```
psfile [УдаленныйКомпьютер] [[Id | путь] [-c]]
```

PsGetSid

```
psgetsid [УдаленныеКомпьютеры] [имя | SID]
```

PsInfo

```
psinfo [УдаленныеКомпьютеры] [-h] [-s] [-d] [-c [-t разделитель]] [поле]
```

PsKill

```
pskill [УдаленныйКомпьютер] [-t] {PID | имя}
```

PsList

```
pslist [УдаленныйКомпьютер] [[-t] | [[-m] [-d] [-x]]] [-s [n] [-r n]] [имя | PID]
```

PsLoggedOn

```
psloggedon [\\компьютер | \\*] [-l] [-x]
```

PsLogList

```
psloglist [УдаленныеКомпьютеры] [-s [-t разделитель] | -x] [-n #] [-r] [-w]
[-a мм/дд/гггг] [-b мм/дд/гггг] [-d #|-h #|-m #] [-f фильтр]
[-i ID[,ID[,...]] | -e ID[,ID[,...]]]
[-o источник[,источник[,...]] | -q источник[,источник[,...]]]
[-z] [-c] [-g имя_файла | -l имя_файла] [журнал_событий]
```

PsPasswd

Для локальных учетных записей:

```
pspasswd [УдаленныеКомпьютеры] ЛокальнаяУчетнаяЗапись [НовыйПароль]
```

Для учетных записей домена:

```
pspasswd Домен\УчетнаяЗапись [НовыйПароль]
```

PsService

```
psservice [УдаленныйКомпьютер] [команда [параметры]]
```

PsService поддерживает следующие команды и параметры:

```
query [-g группа] [-t {driver|service|interactive|all}] [-s
{active|inactive|all}] [служба]
config [служба]
depend служба
security служба
find служба [all]
setconfig служба {auto|demand|disabled}
start служба
stop служба
restart служба
pause служба
cont служба
```

PsShutdown

```
psshutdown [УдаленныеКомпьютеры] {-s|-k|-r|-h|-d|-l|-o|-a} [-f] [-c]
[-t [секунды|чч:мм]] [-v секунды] [-e [u|p] :xx:yy] [-m "сообщение»] [-n
секунды]
```

PsSuspend

```
pssuspend [УдаленныйКомпьютер] [-r] {PID|имя}
```

Системные требования PsTools

В табл. 6-4 перечислены требования каждой из утилит PsTools для локального и удаленного выполнения операций.

Табл. 6-4. Системные требования PsTools

Утилита	Локальные операции	Удаленные операции		
	Требуются администраторские права на локальной системе	Требуется общая папка Admin$ на удаленной системе	Требуется служба Remote-Registry	Поддерживает указание имен нескольких компьютеров
PsExec	Зависит от команды и параметров	Да	Нет	Да
PsFile	Да	Нет	Нет	Нет
PsGetSid	Нет	Да	Нет	Да

Табл. 6-4. (*окончание*)

Утилита	Локальные операции	Удаленные операции		
PsInfo	Нет	Да	Да	Да
PsKill	Зависит от целевого процесса	Да	Нет	Нет
PsList	Нет	Да	Да	Нет
PsLogged-On	Нет	Нет	Да	(Может сканировать сеть)
PsLogList	Зависит от операции и целевого журнала	Да	Да	Да
PsPasswd	Да	Нет	Нет	Да (для локальных учетных записей)
PsService	Зависит от операции и службы	Нет	Нет	Нет (но параметр find может сканировать сеть)
PsShut-down	Да	Да	Нет	Да
PsSuspend	Зависит от целевого процесса	Да	Нет	Нет

Глава 7

Утилиты для работы с процессами и диагностики

Утилиты Process Explorer и Process Monitor (см. главы 3 и 4) — основные утилиты для анализа поведения и динамики состояния процессов и системы в целом. В этой главе мы рассмотрим шесть дополнительных утилит Sysinternals, позволяющих просматривать сведения о состоянии процесса:

- **VMMap** — утилита с графическим интерфейсом, показывает информацию об использовании процессом виртуальной и физической памяти;
- **ProcDump** — консольная утилита, генерирующая дамп памяти процесса с заданным характеристиками, например, процесса резко увеличившего потребление ресурсов процессора или «зависшего»;
- **DebugView** — утилита с графическим интерфейсом, позволяющая просматривать вывод отладчиков, работающих в режиме пользователя или режиме ядра на локальном или удаленном компьютере;
- **LiveKd** — утилита, позволяющая запускать стандартный отладчик ядра на моментальном снимке работающей локальной системы без необходимости перезагружаться в режиме отладки;
- **ListDlls** — консольная утилита, отображающая информацию о загруженных библиотеках DLL;
- **Handle** — консольная утилита, отображающая информацию об описателях объектов, используемых системными процессами.

VMMap

VMMap — утилита для анализа виртуальной и физической памяти, используемой процессом (рис. 7-1). Она показывает в графическом и табличном виде общие сведения о различных типах памяти, выделенных процессу, а также подробные карты распределения виртуальной памяти, отображая такие характеристики, как резервные файлы и типы защиты. VMMap также показывает сводную и подробную информацию о размере физической памяти (рабочего набора), связанного операционной системой с различными блоками виртуальной памяти.

С помощью VMMap можно сделать несколько моментальных снимков памяти процесса и отобразить его изменения в виде графика, чтобы найти все изменения, произошедшие между моментами создания снимков. В сочетании с параметрами фильтрации и обновления это позволяет определять источники использования памяти процесса и расход памяти функциями приложений.

Кроме того, VMMap «впрыскивает» в процессы свою DLL для мониторинга распределения их памяти, путей к коду и стеков вызовов, формирующих данную картину распределения памяти. При наличии полных символов VMMap покажет и строку исходного кода, вызвавшего то или иное изменение в распределении памяти.

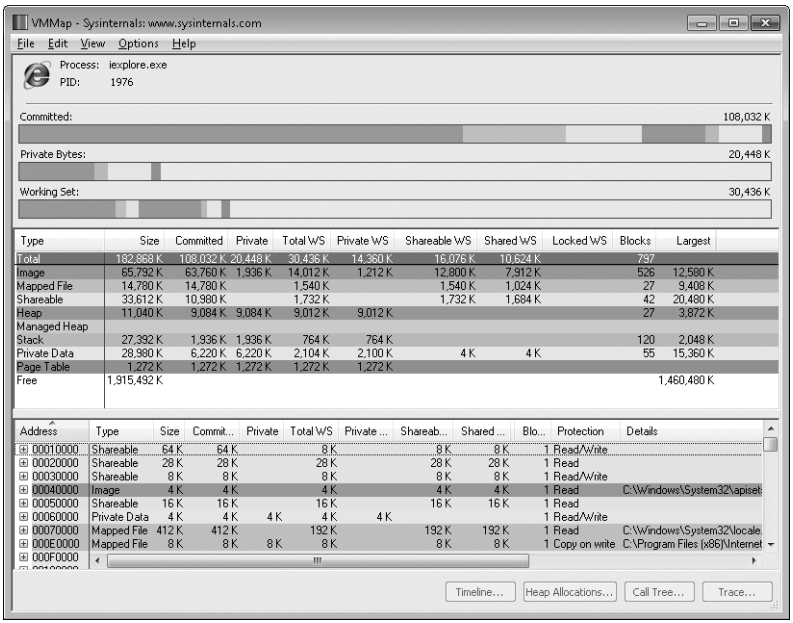


Рис. 7-1. Главное окно VMMap

VMMap поддерживает не только представления с гибкими возможностями для анализа процессов в реальном времени, но и экспорт данных в разных форматах, включая собственный формат, сохраняющий подробные данные и позволяющий загружать их обратно в VMMap для просмотра. Также эта утилита поддерживает параметры командной строки для использования в сценариях.

VMMap — идеальная утилита для разработчиков, желающих понять и оптимизировать использование памяти их приложениями. (Увидеть общесистемное распределение системной памяти в Windows можно с помощью утилиты RAMMap, описанной в главе 14.) VMMap работает в Windows XP (редакции x86 и x64) и выше.

Запуск VMMap и выбор процесса

Первое, что нужно сделать, запустив VMMap, — выбрать процесс для анализа. Если не указать процесс во входном файле или командной строке (см. ниже), VMMap откроет окно **Select or Launch Process (Выбор или запуск процесса)**. На вкладке **View a Running Process (Просмотр запущенного процесса)** можно выбрать уже запущенный процесс, а с вкладки **Launch and Trace a New Process (Запуск и отслеживание нового процесса)** можно запустить новый процесс и следить за его распределениями памяти. Окно **Select or Launch Process (Выбор или запуск процесса)** можно открыть и позже, нажав Ctrl+P.

Просмотр запущенного процесса

Выберите процесс на вкладке **View a Running Process (Просмотр запущенного процесса)** (рис. 7-2) и щелкните **OK**. Чтобы быстро найти процесс по идентификатору (PID) или по использованию памяти, щелкните на любом заголовке столбца, чтобы отсортировать строки по этому столбцу. Доступны следующие столбцы: **User (Пользователь)**, **Private Bytes (Байты закрытой памяти)**, **Working Set (Рабочий набор)** и **Architecture (Архитектура)** (в последнем показано, является ли процесс 32-разрядным или 64-разрядным). Чтобы обновить список, щелкните кнопку **Refresh (Обновить)**.

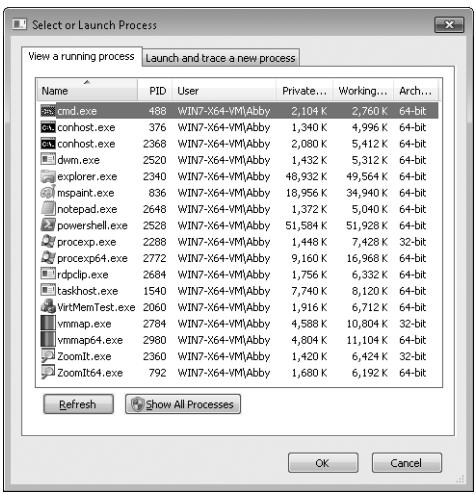


Рис. 7-2. Окно Select or Launch Process со списком запущенных процессов

На вкладке **View a Running Process (Просмотр запущенного процесса)** перечислены только те процессы, которые VMMap может открыть. Если VMMap запущена без администраторских прав (включая привилегию Debug [Отладка]), список будет включать только процессы, запущенные тем же пользователем, что запустил VMMap, на том же или более низком уровне целостности (IL). В Windows Vista и выше можно перезапустить VMMap

с повышенными правами, щелкнув кнопку **Show All Processes (Показать все процессы)** или выбрав команду меню **File | Run As Administrator (Файл | Запустить от имени администратора)**.

В Windows с архитектурой x64 VMMap может анализировать 32- и 64-разрядные процессы. Для анализа 32-разрядных процессов VMMap запускает свою 32-разрядную версию и 64-разрядную для анализа 64-разрядных процессов. (Подробнее см. в главе 1.) Если использовать параметр командной строки -64 (см. ниже), 64-разрядная версия утилиты будет использоваться для анализа всех процессов.

Запуск и мониторинг нового процесса

Если запустить приложение из VMMap, оно получит инструментарий для отслеживания отдельных операций выделения памяти и соответствующих вызовов в стеке. Введите путь к приложению и, если требуется, аргументы командной строки и папку для запуска, как показано на рис. 7-3, после чего щелкните **OK**.

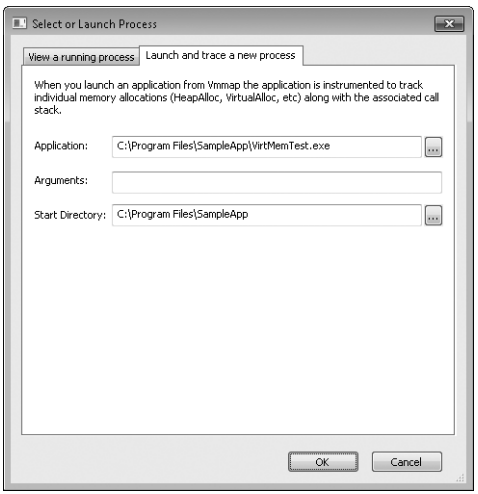


Рис. 7-3. Запуск и отслеживание нового процесса

VMMap подгружает свою DLL в целевой процесс при его запуске и перехватывает его вызовы API виртуальной памяти. Кроме типа выделяемой памяти, размера и сведений о ее защите, VMMap собирает данные о стеке вызовов в момент выделения. VMMap по-разному группирует эти сведения (см. ниже в этой главе).

В Windows с архитектурой x64 VMMap может “впрыскивать” DLL со своим инструментарием и отслеживать программы для x86 и x64, при этом запускается 32- или 64-разрядная версия утилиты, соответственно. Однако в Windows x64 VMMap не может отслеживать программы .NET, собранные для «любого ЦП» («Any CPU»), это возможно только в 32-разрядных версиях Windows. Однако мониторинг возможен и в 64-разрядных версиях

Windows — путем выбора на вкладке **View a Running Process (Просмотр запущенного процесса)** окна **Select or Launch Process (Выбор или запуск процесса)**.

Примечание Архитектура «Any CPU» используется по умолчанию при сборке приложений на C# и Visual Basic .NET в Visual Studio 2005 и выше.

Окно VMMap

После выбора или запуска процесса VMMap анализирует его, показывает графическое представление виртуальной и физической памяти, а также табличные представления **Summary (Общие)** и **Details (Подробно)**. В них типы памяти представлены различными цветами. Представление **Summary** также выделено отдельным цветом.

Первая гистограмма в окне VMMap (рис. 7-1) — **Committed (Переданная)**. Ее разноцветные части показывают относительные пропорции различных типов памяти, преданной в адресное пространство процесса. Также она служит эталоном, с которым сравниваются две другие диаграммы. Общее число, показанное над правой частью диаграммы, показывает не *всю* распределенную память, а только «доступную» процессу память. Области, которые пока только зарезервированы, еще не доступны, а потому не отображаются на этой диаграмме. Другими словами, сюда включается физическая память (ОЗУ), память из файла подкачки и спроецированных в память файлов.

Вторая диаграмма — **Private Bytes (Байты закрытой памяти)**. Здесь отображается память процесса, не разделяемая с другими процессами, выделенная в ОЗУ или файле подкачки. Сюда входит стек, кучи, неформатированная виртуальная память, страничные таблицы и области чтения-записи проекций образов и файлов. Надпись над правой частью диаграммы показывает общий размер собственной памяти процесса. Цветные области диаграммы показывают пропорции различных типов закрытой памяти. Отношение размера цветных областей к общей длине диаграммы показывает их долю в используемой виртуальной памяти.

Третья гистограмма показывает *рабочий набор* процесса. Рабочий набор — это виртуальная память процесса, загруженная в физическую оперативную память. Как и на предыдущей диаграмме, цветные области показывают доли различных типов памяти в ОЗУ, а их размер — относительную величину по сравнению с общей виртуальной памятью, выделенной процессу и загруженной в ОЗУ.

Заметьте, что эти диаграммы показывают только относительные пропорции различных типов памяти, но не являются картами памяти, отображающими ее размещение. Такую карту для 32-разрядных процессов предоставляет окно **Address Space Fragmentation (Фрагментация адресного пространства)**, описанное далее в этой главе.

Под этими диаграммами расположена таблица **Summary View (Сводка)**, в которой указаны разные типы памяти (см. раздел «Типы памяти» ниже),

общий объем каждого типа, объем переданной процессу памяти и объем физического ОЗУ. Можно выбрать тип памяти в этой таблице, чтобы отфильтровать содержимое секции **Details View (Подробные сведения)**. Таблицу можно отсортировать по значениям любого столбца, щелкнув его заголовок. Если щелкнуть тот же заголовок еще раз, данные будут отсортированы в обратном порядке. Порядок столбцов можно изменять, перетаскивая их. Можно также изменять ширину столбцов, растягивая границы между их заголовками.

Секция **Details View (Подробные сведения)**, расположенная под таблицей **Summary View**, содержит информацию о каждой области памяти виртуального адресного пространства процесса, работающего в пользовательском режиме. Чтобы отображался только один тип, выберите этот тип в таблице **Summary View**. Чтобы вернуться к просмотру всех типов памяти, выберите в таблице **Summary View** строку **Total (Всего)**. Сортировку, порядок и ширину столбцов в секции **Details View** можно изменять так же, как и в таблице **Summary View**.

Строки типов памяти в секции **Details View**, можно раскрывать, чтобы увидеть блоки, которыми выделялась память этого типа. Такая ситуация бывает, например, когда резервируется крупный блок памяти и передается процессу по частям. Это происходит и когда загрузчик образа или приложение создает проекцию файла, а затем создает несколько проецированных представлений этой проекции; например, чтобы установить разную защиту для разных областей проекции. Можно раскрывать и сворачивать отдельные группы подчиненных сопоставлений, щелкая на значках (+) и (–) в секции **Details View**. Также можно раскрыть или свернуть все группы сразу, выбрав в меню **Options (Параметры)** команду **Expand All (Раскрыть все)** или **Collapse All (Свернуть все)**. Верхняя строка такой группы показывает суммы компонентов группы. Когда выбирается другой порядок сортировки, блоки остаются в строках верхнего уровня и сортируются внутри группы.

Если вам не нравится шрифт, выбранный для VMMap по умолчанию, можете выбрать другой шрифт для **Summary View**, **Details View** и некоторых диалоговых окон.

Типы памяти

VMMap разделяет выделенную память на несколько типов.

- **Образ (Image)** — такая память представляет исполняемые файлы, такие как EXE и DLL, загруженные в процесс загрузчиком образов. Заметьте, что этот тип не включает исполняемые файлы, загруженные как файлы данных, которые относятся к типу памяти Mapped File (Проецируемый файл). Области исполняемого кода обычно доступны только для чтения и выполнения, возможно их разделение. Области данных, такие как инициализированные данные, обычно доступны для чтения и записи или с копированием при записи. Когда изменяются страницы, доступные с

копированием при записи, процесс получает дополнительную закрытую память, которая помечается как доступная для чтения и записи. Эта закрытая память находится в ОЗУ или файле подкачки, а не в файле образа. В столбце **Details (Подробно)** представления **Details View** показан путь к файлу или имя секции.

- **Спроецированный файл (Mapped File)** — может использоваться совместно и представляет файл на диске. Проецируемые файлы часто являются библиотеками DLL с ресурсами и обычно содержат данные приложений. В столбце **Details (Подробно)** показан путь к файлу.
- **Разделяемая (Shareable)** — такая память может сообща использоваться процессами и находится в ОЗУ или файле подкачки (если он имеется). Эта память обычно содержит данные, совместно используемые процессами посредством общих секций библиотек DLL или объекты, находящиеся в файле подкачки или проецируемом файле (такие файлы называют также секциями в файле подкачки).
- **Куча (Heap)** — закрытая память, распределяемая и управляемая диспетчером кучи пользовательского режима; обычно содержит данные приложений. Функции управления памяти приложений, использующие кучу, включают библиотечную функцию исполняющей среды C *malloc*, оператор C++ *new*, API-функции Windows *Heap* и устаревшие API-функции *GlobalAlloc* и *LocalAlloc*.
- **Управляемая куча (Managed Heap)** — представляет закрытую память, распределяемую и управляемую исполняющей средой .NET; обычно содержит данные приложений.
- **Стек (Stack)** — память в стеке выделяется каждому потоку в процессе для хранения параметров функций, локальных переменных и записей о вызовах. Обычно при создании потока ему выделяется и резервируется фиксированный объем памяти стека, но процессу передается только сравнительно малая ее часть. По мере надобности память из данной выделенной области передается, но не освобождается, пока поток не завершится.
- **Закрытая память (Private Data)** — эта память выделяется API-функцией *VirtualAlloc* и не обрабатывается диспетчером кучи или исполняющей средой .NET и не относится к категории памяти стека. Такая память обычно содержит данные приложений, а также блоки Process и Thread Environment. Закрытые данные не могут совместно использоваться разными процессами.

Примечание Определение «закрытой памяти» в VMMap шире одноименного термина, принятого в Process Explorer: в первом под «закрытой памятью» понимают *всю* закрытую память, переданную процессу.

- **Таблица страниц (Page Table)** — закрытая память режима ядра, связанная с таблицами страниц процесса. Обратите внимание на то, что эта память никогда не отображается в представлении VMMap **Details (Подробно)**, в котором отображается только память пользовательского режима.

- **Свободная (Free)** — свободные области памяти (нераспределенные области виртуального адресного пространства процесса). Чтобы включить свободные области памяти в представление **Details (Подробно)** при просмотре общей карты памяти процесса, выберите команду **Options | Show Free Regions** (Параметры | Показать свободные области).

Информация о памяти

В представлениях **Details** и **Summary** отображается следующая информация о типах памяти и операциях ее распределения. Чтобы уменьшить объем "мусора" в выходных данных, VMMap не отображает записи об операциях с нулевым объемом памяти.

- **Size (Размер)** — общий размер распределенного типа или области. Сюда входят области, которые были зарезервированы, но не переданы процессам.
- **Committed (Переданная)** — объем памяти, переданной процессам, т.е. расположенной в ОЗУ, файле подкачки или проецируемом файле.
- **Private (Закрытая)** — объем памяти, принадлежащей единственному процессу.
- **Total WS (Рабочий набор, всего)** — общий размер рабочего набора (физической памяти), связанного с этим типом памяти.
- **Private WS (Рабочий набор, закрытый)** — размер неразделяемого рабочего набора.
- **Shareable WS (Рабочий набор, общий)** — размер разделяемого рабочего набора.
- **Shared WS (Рабочий набор, совместно используемый)** — размер рабочего набора, разделяемого с другими процессами в текущий момент.
- **Locked WS (Рабочий набор, заблокированный)** — объем памяти, гарантированно остающейся в физической памяти и не вызывающей страничных ошибок при обращении.
- **Blocks (Блоки)** — число областей памяти, выделенных по отдельности.
- **Largest (Наибольший)** — размер наибольшего непрерывного блока памяти для данного типа (в представлении Summary).
- **Address (Адрес)** — базовый адрес области памяти в виртуальном адресном пространстве процесса (в представлении Details).
- **Protection (Защита)** — в представлении Details показывает типы операций, которые можно выполнять с памятью. В случае групп операций, включающих выделение блоков памяти, в столбце **Protection** отображается сводная информация о типах защиты выделенных блоков. При попытке выполнения кода из области, не помеченной флагом Execute (если включен механизм DEP), либо записи в область, не имеющей флага Write или Copy-on-Write, либо при обращении к памяти, помеченной как no-access или зарезервированной, но еще не выделенной, возникает ошибка нарушения доступа.

- **Details (Подробно)** — в представлении Details содержит дополнительную информацию для области памяти, такие как путь к файлу, в котором хранятся страницы, идентификатор кучи (для памяти из кучи), идентификатор потока (для памяти стека) или поколение .NET AppDomain и Garbage Collection.

Примечание API *VirtualProtect* может изменять типы защиты любой страницы, установленный при ее выделении. Это означает, что в разделяемой области памяти могут появиться страницы закрытой памяти (например, если эта область выделена в файле подкачки, после чего приложение сменило тип защиты на copy-on-write, а затем модифицировала эти страницы).

Хронология моментальных снимков

VMMap сохраняет моментальные снимки состояния выделения памяти для целевого процесса. Можно загрузить любой из моментальных снимков в главное окно VMMap и сравнить с другим снимком для поиска изменений.

При отслеживании оснащенного процесса VMMap делает моментальные снимки автоматически. В меню **Options | Trace Snapshot Interval (Параметры | Интервал получения моментального снимка)** можно выбрать интервал в 1, 2, 5 или 10 секунд. Можно приостанавливать и возобновлять автоматическое получение моментальных снимков, нажимая Ctrl+Space, и вручную получать моментальные снимки в любое время нажатием F5.

При анализе процессов, не содержащих библиотеки VMMap, моментальные снимки не создаются автоматически, их приходится делать вручную, нажимая F5.

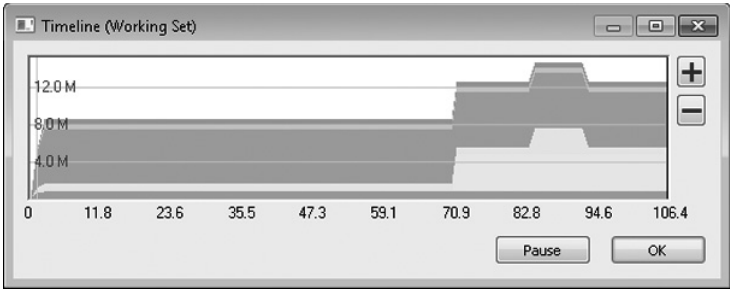


Рис. 7-4. Окно Timeline в VMMap

Щелкните кнопку **Timeline (Хронология)** в основном окне VMMap, чтобы открыть одноименное окно (рис. 7-4) с графическим представлением истории выделения памяти в рабочем наборе процесса. Инструмент Timeline позволяет загружать ранее полученные моментальные снимки в основное окно VMMap и попарно сравнивать их. Горизонтальная ось графика представляет число секунд с момента получения первоначального моментального снимка, а вертикальная ось — обращения к рабочему набору процесса. Цвета на графике соответствуют цветам, обозначающим типы памяти в основном окне VMMap.

Когда включено автоматическое получение моментальных снимков для отслеживания процесса, содержимое диалогового окна **Timeline** автоматически обновляется. Можно щелкнуть кнопку **Pause (Пауза)**, чтобы приостановить автоматическое получение моментальных снимков; и щелкнуть еще раз, чтобы возобновить его. Чтобы обновить окно **Timeline** при просмотре процесса, не содержащего DLL VMMap, нужно закрыть это окно и снова открыть его.

Щелкните в любой точке хронологии, чтобы загрузить в основное окно VMMap соответствующий моментальный снимок. Чтобы сравнить любые два моментальных снимка, щелкните точку под одним из снимков и перетащите указатель мыши ко второй точке. Пока кнопка мыши нажата, на временной шкале отображаются вертикальные линии, показывающие, моменты получения моментальных снимков, а область между двумя выбранными точками затеняется (рис. 7-5). Чтобы изменить детализацию временной шкалы для облегчения выбора моментальных снимков, щелкните кнопки со значками (+) и (−), затем перетащите горизонтальный ползунок прокрутки.

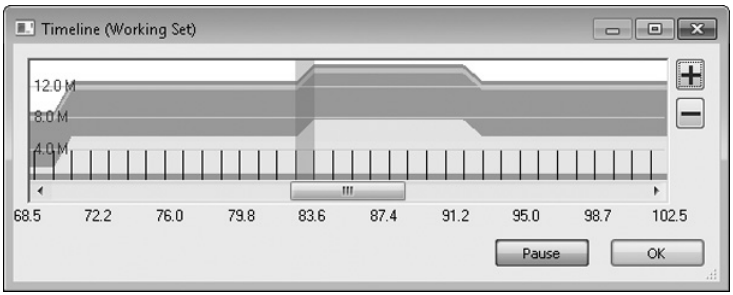


Рис. 7-5. Окно VMMap Timeline во время выбора двух моментальных снимков

При сравнении моментальных снимков диаграммы в основном окне VMMap отображают различия между этими снимками в виде прироста или убыли значений с момента получения предыдущего моментального снимка. Диапазоны адресов в представлении Details, имеющиеся в новом моментальном снимке, но отсутствующие в предыдущем, выделяются зеленым, а диапазоны, имеющиеся только в предыдущем снимке, — красным. Чтобы увидеть эти диапазоны, раскройте соответствующие узлы дерева операций выделения памяти. Строки представления Details, не изменившие цвет, показывают изменения размера рабочего набора. Для просмотра изменений определенного типа памяти выберите этот тип в представлении Summary.

Если выбрать в меню **View (Вид)** команду **Empty Working Set (Очистить рабочий набор)**, VMMap освободит всю физическую память, назначенную процессу, и получит новый моментальный снимок. Эта функция полезна для измерения памяти, потребляемой отдельными функциями, которые вызывают приложения: опустошите рабочий набор, вызовите функцию и обновите экран, чтобы узнать, сколько физической памяти ушло у приложения на этот вызов.

Чтобы переключиться со сравнения на просмотр одного моментального снимка, откройте окно **Timeline (Временная линейка)** и выберите любой моментальный снимок.

Просмотр текста из областей памяти

В некоторых случаях предназначение области памяти можно понять по хранящимся в ней строкам. Для просмотра строк длиной от трех символов ASCII или Unicode, выберите область в представлении Details, а затем выберите команду меню **View | Strings (Вид | Строки)**. VMMap откроет окно, показывающее диапазон виртуальных адресов и найденные в нем строки (рис. 7-6). Если внутри выбранной области есть блоки, будет выполнен поиск по всем блокам в области.

Строки данных не входят в моментальные снимки. Эта функция работает только с выполняющимися процессами, но не с сохраненными файлами VMMap (.mmp-файлами), загруженными с диска. Кроме того, строки считываются напрямую из памяти процесса при вызове этой функции, так что они могут измениться с момента получения последнего моментального снимка.

Примечание В программировании термин «строка» обозначает структуру данных, состоящую из последовательности символов, обычно представляющую понятный человеку текст.

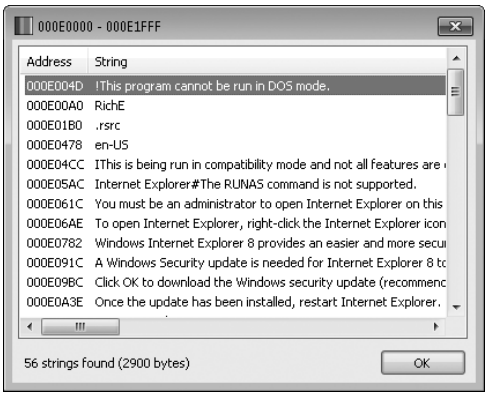


Рис. 7-6. Окно VMMap Strings

Поиск и копирование текста

Чтобы найти определенный текст в представлении Details, нажмите Ctrl+F. Функция поиска подсветит следующую видимую строку представления Details, содержащую указанный текст в любом столбце. Заметьте, что поиск текста внутри свернутых областей с блоками не выполняется. Чтобы повторить предыдущий поиск, нажмите F3.

VMMap предоставляет два способа копирования текста в буфер обмена:

- нажатие Ctrl+A копирует весь текст из окна VMMap, включая имя и идентификатор процесса, и весь текст из представлений Summary и De-

tails с сохранением порядка сортировки. Копируются все данные о вложенных в группу операциях выделения, даже если группа свернута. Если в представлении Summary выделен определенный тип памяти, из представления Details будут скопированы только данные об этом типе;

- нажатие Ctrl+C копирует весь текст из таблицы Summary, если активно представление Summary. Если в фокусе находится представление Details, будет скопировано поле адреса из выбранной строки, которое затем можно вставить в отладчик.

Просмотр распределения памяти в процессе с подгруженной DLL VMMap

Подгрузив в процесс свою DLL, VMMap перехватывает вызовы API виртуальной памяти и получает следующую информацию о них:

- имя функции, указывающее тип выделяемой памяти. Например, *VirtualAlloc* и *VirtualAllocEx* выделяют закрытую память; *RtlAllocateHeap* выделяет память в куче;
- операция, такая как Reserve, Commit, Protect (изменение защиты) и Free;
- тип защиты памяти, такой как «выполнение/чтение» и «чтение/запись»;
- запрошенный размер в байтах;
- адрес виртуальной памяти, по которому выделен блок;
- стек вызовов в момент вызова API.

Стек вызовов показывает путь к коду, инициировавшему запрос выделения памяти. VMMap назначает каждому уникальному стеку вызовов номер (Call Site ID). Первый стек вызовов получает номер 1, второй — номер два и т.д. Если некоторый код выполняется несколько раз, у каждого вызова будет один и тот же стек, и данные о выделенной этими вызовами памяти будут сгруппированы под одним номером Call Site ID.

Примечание Для получения полезной информации о процессах без подгруженной DLL необходима корректная настройка символов (см. главу 2).

Обновите основное окно VMMap и щелкните кнопку **Trace (Трассировка)**. Откроется окно **Trace (Трассировка)** со списком всех операций выделения памяти, сгруппированных по идентификатору вызывающего кода (рис. 7-7). В столбце **Function (Функция)** указана вызванная API-функция; в столбце **Calls (Вызовы)** указано, сколько раз она была вызвана; в столбце **Bytes (Байты)** — общий объем памяти, выделенной этим кодом. Столбцы **Operation (Операция)** и **Protection (Защита)** содержат значения, переданные при первом вызове этой функции.

Щелкните знак +, чтобы раскрыть узел вызовов и просмотреть адреса виртуальной памяти, по которым предоставлена запрошенная память. В столбце **Bytes (Байты)** показан размер каждой выделенной области. Заметьте: если память освобождается, последующие запросы выделения, отправленные тем же кодом, могут удовлетворяться выделением памяти

по тому же адресу. Когда это происходит, VMMap не регистрирует новую запись, а выводит в столбце **Bytes (Байты)** размер первой области, выделенной по этому адресу, но суммарный размер памяти для данной функции подсчитывается правильно.

По умолчанию окно **Trace (Трассировка)** показывает только те операции, для которых значение в столбце **Bytes (Байты)** больше нуля. Чтобы отображать и «нулевые» операции, установите флажок **Show All Memory Operations (Показывать все операции с памятью)**. Сюда входят такие операции, как *RtlCreateHeap*, *RtlFreeHeap* и *VirtualFree* (при освобождении целого блока).

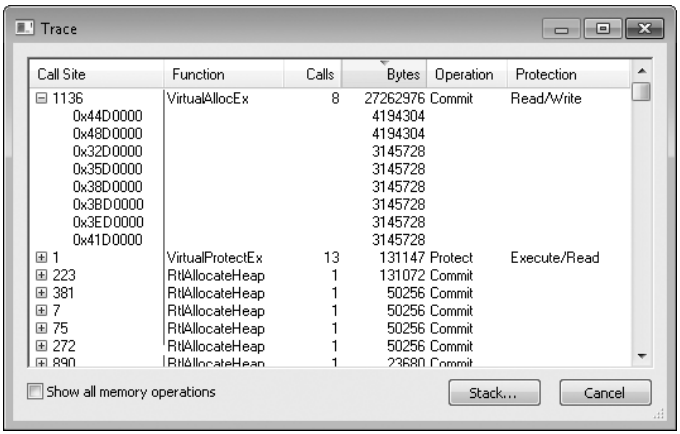


Рис. 7-7. Окно VMMap Trace

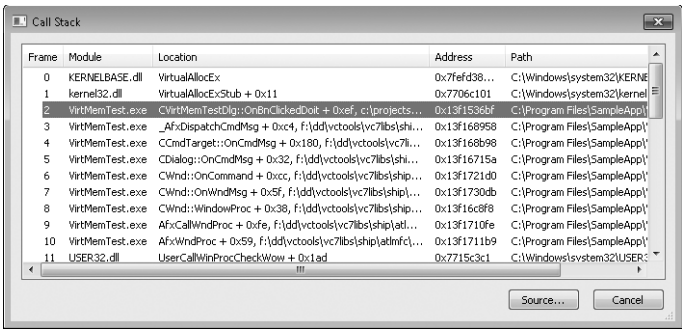


Рис. 7-8. Стек вызовов для функции, открытый из окна Trace

На рис. 7-7 код с ID 1136 был вызван восемь, в результате выделено 26 МБ закрытой памяти. Этот узел раскрыт, видны адреса виртуальной памяти и размеры запрошенной памяти. Поскольку все эти запросы отданы одним кодом, можно выбрать любой из них или узел верхнего уровня и щелкнуть кнопку **Stack (Стек)**, чтобы просмотреть стек вызовов для этой функции (рис. 7-8). Если доступны полные символы и исходный текст, выберите в стеке вызовов фрейм и щелкните кнопку **Source (Исходный код)**, чтобы

просмотреть исходный код при помощи утилиты просмотра исходных файлов VMMap, которая выделит соответствующую строку в коде.

Щелкните кнопку **Call Tree (Дерево вызовов)** в главном окне VMMap, чтобы получить еще одно представление выделения памяти. Окно **Call Tree (Дерево вызовов)** (рис. 7-9) показывает сходства и различия всех записанных стеков в виде дерева. Верхние узлы представляют самые давние вызовы в стеке. Их дочерние узлы представляют функции, вызываемые ими, а «вложенные» узлы — функции, вызываемые в ходе операций с памятью. Столбцы **Count (Счет)** и **% Count (Счет %)** в каждой строке показывают, сколько раз был вызван этот код во всех стеках; а столбцы **Bytes (Байты)** и **% Bytes (Байты %)** показывают, сколько памяти было этими вызовами. Эти данные можно использовать для быстрого кода, выделяющего максимальный объем памяти или отправляющего наибольшее число запросов на выделение.

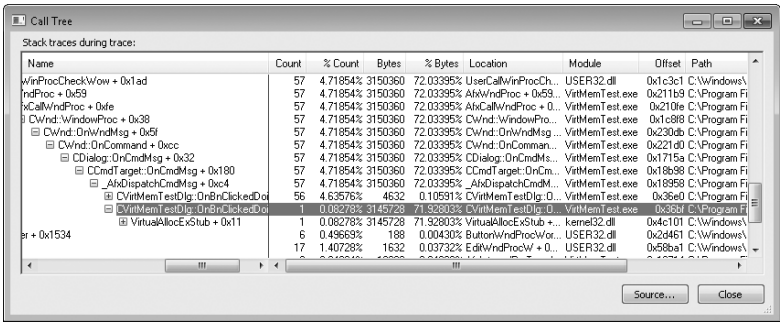


Рис. 7-9. Окно VMMap Call Tree

Наконец, можно увидеть стек вызовов для определенной операции выделения памяти в куче, выбрав ее в представлении Details и щелкнув кнопку **Heap Allocations (Выделение из кучи)**, чтобы открыть одноименное окно (рис. 7-10). Выберите элемент в этом окне и щелкните кнопку **Stack (Стек)** — откроется стек вызовов для этой операции выделения памяти.

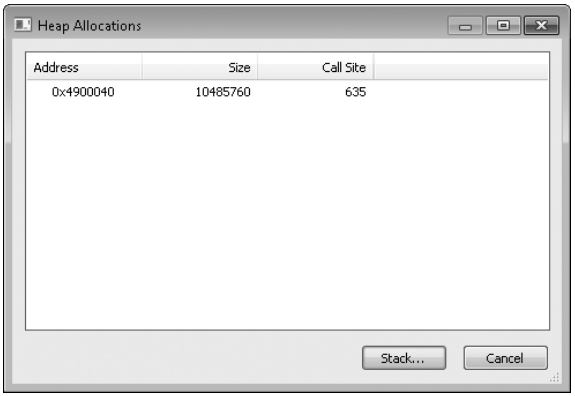


Рис. 7-10. Окно Heap Allocations

Фрагментация адресного пространства

Некорректное управление памятью может привести к тому, что при большом объеме свободной памяти не окажется ни одного свободного блока, достаточного для удовлетворения очередного запроса на выделение памяти. Окно **Address Space Fragmentation (Фрагментация адресного пространства)** показывает компоновку различных типов областей памяти, выделенной в адресном пространстве 32-разрядного процесса (рис. 7-11). Эта информация помогает выяснить, является ли источником проблемы фрагментация, и найти области, создающие ее.

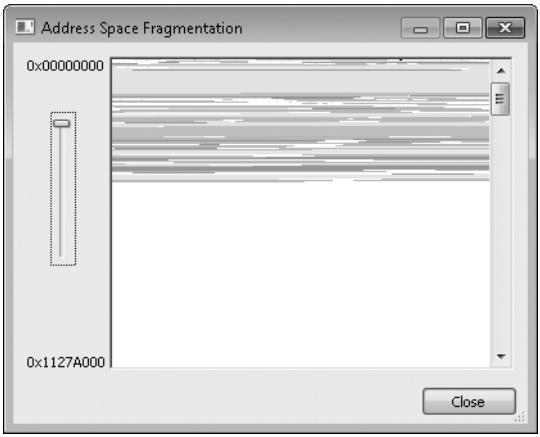


Рис. 7-11. Фрагментация адресного пространства (только для 32-разрядных процессов)

При анализе 32-разрядного процесса выберите команду меню **View | Fragmentation (Вид | Фрагментация)**, чтобы открыть окно **Address Space Fragmentation (Фрагментация адресного пространства)**. Диаграмма в этом окне показывает типы выделенной памяти, используя те же цвета, что и главное окно VMMap; меньшие виртуальные адреса собраны в верхней части окна. Адреса вверху и слева внизу представляют границы текущего отображаемого диапазона адресов. Если диапазон адресов не умещается в окно, используйте вертикальную полосу прокрутки. Ползунок слева от диаграммы позволяет изменять ее детализацию: перетаскивание его вниз увеличивает размер блоков на диаграмме. Если щелкнуть область на диаграмме, под диаграммой будет показан адрес этой области, ее размер и тип, соответствующая область также будет выделена в представлении Details в главном окне VMMap.

Сохранение и загрузка моментальных снимков

Команды **Save (Сохранить)** и **Save As (Сохранить как)**, расположенные в меню **File (Файл)**, позволяют сохранять моментальный снимок VMMap в разных форматах:

- **.MMP** — собственный формат файлов VMMap. Используйте его, если нужно загрузить снимок в VMMap на том же или другом компьютере.

В этом формате сохраняются все снимки, что позволяет сравнивать их в окне **Timeline** после загрузки в VMMap;

- **.CSV** — данные последнего снимка в виде значений, разделенных запятыми, идеально для импорта в Microsoft Excel. Если в представлении Summary выбран определенный тип памяти, будут сохранены сведения только о нем;
- **.TXT** — форматированный текст, удобен для публикации результатов в понятной человеку форме с использованием моноширинного шрифта. Как и при использовании формата CSV, при выборе определенного типа памяти сохраняются данные только о нем.

Чтобы загрузить в VMMap сохраненный .MMP-файл, нажмите Ctrl+O или укажите имя файла в командной строке, используя параметр –o. Кроме того, при запуске VMMap расширение .mmp сопоставляется с этой программой, что позволяет открывать сохраненные .mmp-файлы двойным щелчком в Проводнике.

Параметры командной строки VMMap

VMMap поддерживает следующие параметры командной строки:

```
vmmap [-64] [-p {PID | имя_процесса} [выходной_файл]] [-o входной_файл]
```

Параметр –64

В версиях Windows x64 VMMap запускает свою 32- или 64-разрядную версию, когда выбирается 32- или 64-разрядный процесс, соответственно. Параметр -64 позволяет использовать 64-разрядную версию VMMap для анализа всех процессов. При анализе 32-разрядных процессов 32-разрядная версия VMMap точнее разделяет типы памяти по категориям. Преимущество 64-разрядной версии в том, что она может определить ID потока, связанный с 64-разрядными стеками, и сообщает более точные данные о системной памяти.

Примечание Параметр -64 применяется только для доступа к работающим процессам; он не действует при впрыскивании DLL и отслеживании процессов, запущенных из VMMap.

Параметр –p

Такой формат используется для анализа процесса, указанного по идентификатору (PID) или имени. Если указать имя, VMMap сопоставит его с первым процессом, имя которого начинается с указанного текста.

Если указать выходной файл, VMMap просканирует целевой процесс, выведет результаты в указанный файл и завершит работу. Если не указать расширение файла, VMMap добавит расширение .MMP и сохранит данные в собственном формате. Если указать расширение .CSV, данные будут сохранены в виде значений, разделенных запятыми. Если указать любое другое расширение, выходные данные будут сохранены в формате TXT.

Параметр –o

Если использовать эту команду, VMMap при запуске откроет указанный входной файл .MMP.

Восстановление параметров VMMap по умолчанию

VMMap сохраняет все параметры конфигурации в разделе реестра HKEY_CURRENT_USER\ Software\Sysinternals\VMMap. Самый простой способ восстановить все параметры конфигурации VMMap, установленные по умолчанию, — закрыть VMMap, удалить этот раздел реестра и снова запустить VMMap.

ProcDump

ProcDump позволяет наблюдать за процессом и создавать дамп-файл в заданные моменты, например, в случае превышения процессом пороговых значений использования ЦП или памяти, возникновения исключений, аварийного завершения, зависания пользовательского интерфейса или превышения пороговых значений счетчиков производительности. ProcDump может делать дамп один раз либо каждый раз, когда возникает заданная ситуация. Кроме того, ProcDump может генерировать дамп сразу либо делать это периодически.

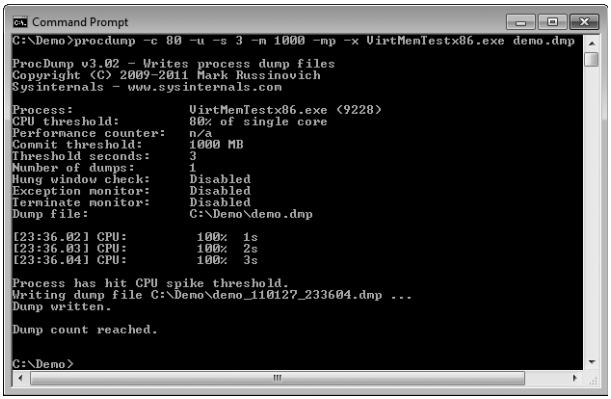


Рис. 7-12. ProcDump запускает процесс и сбрасывает его дамп, если процесс занимает ЦП более трех секунд

Файл дампа процесса — это подробный моментальный снимок внутреннего состояния процесса, который администраторы и разработчики могут использовать для определения источников проблем с приложениями. Файлы дампа анализируются при помощи отладчика, например, WinDbg из средств Debugging Tools for Windows.

ProcDump мало влияет на систему при мониторинге процесса и поэтому является идеальным инструментом для получения информации о проблемах, которые трудно изолировать и воспроизвести, даже если проблемная

ситуация повторяется раз в месяц. ProcDump не останавливает процесс, который он отслеживает, дампы можно получать без вмешательства в работу процесса.

В ProcDump появился новый тип дампов — miniplus, который удобен для громоздких процессов, таких как Exchange Server и SQL Server. Дамп miniplus идентичен полному дампу памяти за исключением крупных областей памяти (таких как кеш). Он позволяет уменьшить размер дампов указанных процессов на 50-90% без ущерба для эффективности анализа (рис. 7-12).

Синтаксис командной строки

Ниже приведен полный синтаксис командной строки ProcDump (см. табл. 7-1); подробнее эти параметры рассматриваются в последующих разделах.

```
procdump [-c процент [-u]] [-s n] [-n счет] [-m предел] [-h] [-e [1] [-b]] [-t]
[-p счетчик порог]
[-ma | -mp] [-r] [-o] [-64]
{ {имя_процесса | PID} [файл_дампа] | -x {файл_образа} {файл_дампа}
[аргументы]}
```

Табл. 7-1. Параметры командной строки ProcDump

Параметр	Описание
Целевой процесс и файл дампа	
имя_процесса	Имя целевого процесса. Экземпляр должен быть уникальным и уже запущенным
PID	Идентификатор (PID) целевого процесса
файл_дампа	Имя файла дампа. Если процесс уже запущен, указывать не обязательно; если используется параметр –x — обязательно
файл_образа	Имя исполняемого файла для запуска
аргументы	Необязательные аргументы командной строки, передающиеся новому процессу
Условия получения дампа	
-c *процент*	% использования ЦП, при превышении которого будет получен дамп
-u	Используется вместе с –c для масштабирования порога с учетом числа процессоров
-s *n*	При использовании с –c указывает длительность высокой загрузки ЦП до получения дампа. При использовании с –p указывает длительность превышения порога счетчика производительности до получения дампа. При использовании только с –n и никакими другими условиями указывает интервал получения дампов в секундах

Табл. 7-1. (*окончание*)

Параметр	Описание
-m *предел*	Указывает предел выделения памяти в МБ, при достижении которого записывается дамп
-h	Создает дамп, когда обнаруживается зависшее окно
-e	Создает дамп, когда возникает необрабатываемое исключение. Если указать значение 1, дамп будет создаваться также при first-chance exception
-b	Если используется вместе с –e, точки останова считаются исключениями; в противном случае игнорируются
-t	Создает дамп при завершении процесса
-p *счетчик порог*	Создает дамп, когда указанный счетчик производительности превышает указанный порог
Параметры файла дампа	
-ma	Включает в дамп память всех процессов
-mp	Создает дамп без крупных областей памяти (miniplus)
-r	Клонирует процесс для получения дампа, чтобы уменьшить время его простоя (только в Windows 7, Windows Server 2008 R2 и выше)
-o	Переписывает существующий файл дампа
-64	Создает 64-разрядный дамп (только для версий 64-разрядных версий Windows)

Выбор процесса для мониторинга

Можно запустить целевой процесс из командной строки ProcDump или отслеживать уже запущенный процесс. Чтобы запустить процесс при помощи ProcDump, используйте параметр –x с именем исполняемого файла, именем файла для записи дампа и аргументами командной строки, которые нужно передать запускаемой программе. Заметьте, что необходимо указать реальное имя исполняемого файла, т.к. ProcDump не сможет вызвать приложение по его отображаемому имени. Параметр –x и следующие за ним параметры должны стоять в конце командной строки ProcDump.

Для мониторинга уже запущенной программы укажите в командной строке имя образа или идентификатор процесса (PID). Если существует несколько процессов с указанным именем, ProcDump не выберет ни один из них; в этом случае следует указать PID.

Для мониторинга процесса, запущенного в том же контексте безопасности, что и ProcDump, администраторские права не требуются. Для мониторинга приложения, запущенного от имени другого пользователя или с более высоким уровнем целостности требуются администраторские права, включая привилегию Debug (Отладка).

Указание пути к файлу дампа

Параметр командной строки файл_дампа указывает путь к файлу дампа и его имя. Этот параметр необходимо указывать при запуске целевого процесса при помощи параметра –x. При выполнении мониторинга уже запущенного процесса этот параметр не является обязательным; если опустить его, ProcDump создаст файл дампа в текущей папке, назвав файл по имени целевого процесса.

Параметр файл_дампа может задавать абсолютный или относительный путь. Если этот путь указывает на существующую папку, ProcDump создаст файл дампа в этой папке, используя имя процесса в качестве имени файла дампа (в противном случае, последняя часть параметра становится базовым именем файла дампа). Например, если указать в качестве параметра файл_дампа путь C:\dumps\sample, и C:\dumps\sample является существующей папкой, ProcDump создаст в этой папке файл дампа с именем процесса. Если такой папки не существует, ProcDump создаст в папке C:\dumps файл дампа с именем sample. Целевая папка должна существовать, иначе ProcDump сообщит об ошибке и завершится.

Чтобы избежать случайной перезаписи других файлов дампа, ProcDump создает уникальные имена файлов дампа, добавляя текущую дату и время в формате базовоеимя_ггММдд_ЧЧммсс.dmp. Например, следующая команда незамедлительно создает файл дампа для Textapp.exe:

```
procdump testapp
```

Если этот дамп был создан в 23:45:56 28 декабря 2010 г., он получит имя *Testapp_101228_234556.dmp*. Такая система именования позволяет получать имена, для которых алфавитный порядок сортировки соответствует хронологическому (для файлов, созданных в 2000—2099 годах). Заметьте, что формат имен файлов фиксирован и не зависит от региональных параметров. ProcDump также устанавливает для файлов дампа расширение *.dmp*.

Дата и время не добавляются в имя файла, только если пользователь создает дамп сразу и указывает для него имя. Например, следующая команда создает файл дампа для приложения Testapp.exe в файле c:\dumps\dumpfile.dmp (если не существует папка c:\dumps\dumpfile):

```
procdump testapp c:\dumps\dumpfile
```

Если файл c:\dumps\dumpfile.dmp уже существует, ProcDump не перепишет его, если не добавить в командную строку параметр -o.

Дампы, которые будут создаваться в дальнейшем при соответствии заданным условиям, всегда будут помечены датой и временем в имени файла.

Указание условий для создания дампа

Как уже говорилось, чтобы получить немедленный дам запущенного процесса, просто укажите его имя или PID без других параметров (имя файла дампа указать можно).

ProcDump может наблюдать за использованием ЦП процессом и создавать дамп при превышении определенного порога в течение указанного промежутка времени. В следующем примере ProcDump создает дамп и завершает работу, если приложение Testapp занимает процессор более чем на 90% в течение пяти секунд:

```
procdump -c 90 -s 5 testapp
```

Если параметр –s не указан, по умолчанию используется промежуток в 10 секунд. Чтобы получить несколько дампов, на случай, если первое превышение вызвано случайными причинами, используйте параметр –n, чтобы указать, сколько дампов необходимо сделать перед завершением работы. В следующем примере ProcDump продолжит мониторинг Testapp и будет создавать новый файл дампа каждый раз, когда приложение займет ЦП более чем на 95% в течение 2 секунд, пока не создаст 10 дампов:

```
procdump -c 95 -s 2 -n 10 testapp
```

В многоядерных системах один поток не может потреблять 100% процессорного времени. Так, в двуядерной системе один поток может потреблять максимум 50%; на четырехядерной — максимум 25%. Чтобы изменить порог –c в соответствии с числом ЦП в системе, добавьте параметр –u. Если выполнить команду procdump –c 90 –u testapp на двуядерной системе, дамп будет создан, когда Testapp будет использовать более 45% ЦП в течение 10 секунд, что эквивалентно 90% на одном из ЦП. Соответственно, в 16-ядерной системе порогом будет значение 5,625%. Поскольку параметр –c требует целого значения, в многоядерных системах применяется параметр –u, повышающий точность заданного порога. Пример использования см. в главе 17.

Примечание Поток пользовательского режима, получающий мало времени ЦП, может исполняться на нескольких ЦП, если только он не привязан к одному ЦП. Параметр –u масштабирует порог лишь по числу ядер, но не гарантирует создание дампа при превышении порога на одном конкретном ЦП. Это все равно невозможно, так как в Windows нет средств для мониторинга соответствующей информации.

Для периодического создания дампов используйте параметры –s и –n без иных условий. Параметр –s указывает число секунд между концом предыдущего дампа и началом следующего. Параметр –n указывает, сколько дампов требуется получить. В следующем примере создается три дампа Testapp: первый незамедлительно, другой через пять секунд и последний еще через пять секунд.

```
procdump -s 5 -n 3 testapp
```

Чтобы создать дамп, когда в процессе возникает необрабатываемое исключение, используйте параметр –e. Параметр –e 1 позволяет создавать дамп при любом исключении, включая первые экземпляры исключений. Параметр –t создает дамп при завершении процесса. Он полезен для определения причины неожиданного завершения процесса, не вызванного необ-

рабатываемым исключением. Если добавить параметр –b, ProcDump будет обрабатывать точки останова как исключения; в противном случае они игнорируются. Например, программа может содержать следующий код:

```
if (IsDebuggerPresent())
    DebugBreak();
```

ProcDump подключается к целевой программе как отладчик, API IsDebuggerPresent возвращает TRUE, и вызывается DebugBreak. ProcDump будет создавать дамп при вызове DebugBreak только в том случае, если указан параметр –b.

Параметр –h создает дамп, когда окно верхнего уровня целевого процесса зависает (перестает «отвечать»). В ProcDump «не отвечает» означает то же, что и в Диспетчере задач Windows: если окно, принадлежащее процессу, не отвечает на оконные сообщения в течение пяти секунд, оно считается зависшим. Для использования этого параметра ProcDump должна работать на том же рабочем столе, что и целевой процесс.

Для получения дампа можно использовать порог выделения памяти процесса. Укажите параметр –m и порог в МБ. В следующем примере дамп создается, когда выделение памяти приложению Testapp превышает 200 МБ:

```
procdump -m 200 testapp
```

ProcDump проверяет счетчики памяти процесса раз в секунду и создает дамп только в том случае, если объем выделенной процессу памяти, сопоставленный с системным пределом выделения (сумма размеров файлов подкачки и большей части ОЗУ), превышает порог в момент проверки. Если произошло кратковременное выделение памяти, ProcDump может не заметить его.

Наконец, в качестве условия создания дампа можно использовать любой счетчик производительности. Укажите параметр –p, имя счетчика производительности и пороговое значение. Если имя счетчика содержит пробелы, заключите его в двойные кавычки. В следующем примере создается дамп Taskmgr.exe, когда число процессов в системе превышает 750 в течение трех секунд:

```
procdump -p «\System\Processes» 750 -s 3 taskmgr.exe
```

Чтобы получить имена счетчиков производительности, можно добавить их в Монитор производительности (Performance Monitor), а затем просмотреть имена на вкладке Данные (Data) в диалоговом окне Свойства (Properties). Однако Perfmon для различения имен экземпляров одного процесса по умолчанию использует нотацию со знаком # и порядковым номером (например, cmd#2), которая непредсказуема и нестабильна: имя, связанное с определенным процессом может измениться, когда запускается или завершается другой процесс. ProcDump не поддерживает такую нотацию, вместо которой использует нотацию процесс_PID, описанную в статье базы знаний

Microsoft Knowledge Base 281884. Например, если имеется два экземпляра Testapp с PID 1135 и 924, к первому из них можно обратиться, указав имя **testapp_1135**. В следующем примере создается дамп этого процесса, если счетчик его описателей превышает 200 в течение трех секунд:

```
procdump -p «\Process(testapp_1135)\Handle Count» 200 -s 3 1135
```

Нотация *процесс_PID* не является обязательной. Можно указать только имя процесса, но если запущено несколько экземпляров этого процесса, результаты будут непредсказуемыми.

Параметры можно комбинировать. Следующая команда создает дамп, если Testapp превышает порог ЦП или выделения памяти, имеет зависшее окно или необрабатываемое исключение, или завершает работу по другим причинам:

```
procdump -m 200 -c 90 -s 3 -u -h -t -e testapp
```

Чтобы прекратить мониторинг, нажмите Ctrl+C или Ctrl+Break.

Параметры файла дампа

В зависимости от того, какую версию библиотеки dbghelp.dll использует ProcDump, доступны разные параметры дампов отладки. Чтобы получить доступ ко всем самым полезным функциям, установите последнюю версию утилит отладки для Windows (Debugging Tools for Windows), скопируйте ProcDump.exe в папку, содержащую файл dbghelp.dll и запустите ее оттуда.

Дампы, созданные ProcDump, содержат, как минимум, базовую информацию о процессе и всех его потоках, включая пути стека для всех потоков, разделы данных из всех загруженных модулей, включая глобальные переменные, и информацию о сигнатурах модулей, чтобы соответствующие файлы символов могли быть загружены с сервера символов, даже если дамп анализируется на совершенно другой платформе.

Примечание При использовании Dbghelp.dll версии 6.1 и выше ProcDump добавляет сведения об использовании ЦП потоками, так что команда отладчика !runaway может показывать объем времени, затраченный каждым потоком. Версия 6.1 входит в состав Windows 7 и Windows Server 2008 R2.

Чтобы включить в дамп всю доступную память процесса, добавьте в командную строку параметр –ma. Если используются новые версии dbghelp.dll, этот параметр также получает информацию об областях памяти, включая сведения о параметрах распределений и защиты. Заметьте, что параметр –ma делает дамп намного больше, а запись памяти крупных приложений на диск может занимать несколько минут. (Дампы miniplus, описанные в следующем разделе, так же полезны, как полные дампы, но на 50–90% меньше их.)

Обычно ProcDump требует приостановки целевого процесса во время получения дампа. В Windows 7 и Windows Server 2008 R2 появилась функция *отражения процесса*, позволяющая «клонировать» процесс, так что он может продолжать работу, пока создается моментальный снимок памяти.

Чтобы использовать эту функцию, добавьте параметр –r. ProcDump создает три файла: файл_дампа.dmp, в который записывается информация о процессе и потоке; файл_дампа-reflected.dmp, в который записывается память процесса; и файл_дампа.ini, связывающий предыдущие два файла и являющийся файлом, который следует открывать в отладчике. Windbg воспринимает *.ini как допустимый тип файлов дампа, хотя в диалоговом окне открытия файлов это не указано.

В версиях x64 ProcDump создает 32-разрядный фал дампа, если целевой процесс является 32-разрядным. Чтобы создать 64-разрядный файл дампа, добавьте в командную строку ProcDump параметр -64.

Дампы miniplus

Тип дампов miniplus (-mp) разработан специально для решения проблемы получения полных дампов крупных приложений, таких как Microsoft Exchange Information Store (store.exe) на крупных серверах. Например, получение полного дампа Exchange 2010 может занять 30 минут и привести к созданию файла размером в 48 ГБ. Сжатие этого файла до 8 ГБ может занять еще 60 минут, а передача сжатого файла в службу поддержки Microsoft займет еще 6 часов. Получение такого же дампа miniplus займет одну минуту, будет получен файл дампа размером в 1,5 ГБ, который можно сжать за 2 минуты и отправить в службу поддержки за 15 минут.

Хотя первоначально этот алгоритм создавался для Exchange, он универсален и работает также для SQL Server и любого другого приложения Microsoft, выделяющего крупные области памяти, благодаря эвристическому поиску данных для включения в дамп.

При получении дампа miniplus сначала создается мини-дамп, к которому добавляется (отсюда «плюс» в названии) память, потенциально полезная для отладки. На первом этапе алгоритм рассматривает только страницы, помеченные как доступные для чтения и записи. Это позволяет исключить большинство страниц образа, но оставляет страницы образа, связанные с глобальными переменными. На следующем этапе алгоритм ищет самую большую область памяти, доступную для чтения/записи, размером более 512 МБ. Если таковая найдена, она временно исключается. Областью считается набор выделенных блоков памяти одинакового размера. Например, если есть двадцать областей по 64 МБ (1280 МБ в сумме) и пять областей по 128 МБ (640 МБ в сумме), будут исключены области по 64 МБ, так как они занимают больше памяти, чем области по 128 МБ, хотя размер этого блока меньше. Эти исключенные области еще могут быть включены. Они делятся на участки по 4 МБ, и если на какой-то из этих участков есть ссылка из какого-либо стека потоков, этот участок включается в дамп.

Даже если процесс не слишком велик, дампы miniplus все равно будут значительно меньше полных дампов, так как не содержат исполняемый образ процесса. Например, полный дамп Блокнота (Notepad) занимает примерно 50 МБ, а дамп miniplus — всего около 2 МБ. Полный дамп Word обыч-

но занимает около 280 МБ, а дамп miniplus — около 36 МБ. Если процесс не слишком большой, можно узнать примерный размер дампа по значению параметра Total/Private в VMMap.

Примечание При отладке дампов miniplus отладчик требует подстановки пропущенных страниц образа из хранилища символов (.sympath) или хранилища исполняемых файлов (.exepath). Если вы создаете дампы miniplus для собственного приложения, вам потребуются символы и исполняемые файлы этого приложения.

Дополнительное преимущество дампов miniplus — возможность восстановления после сбоев чтения памяти. Сбои чтения памяти — причина, по которой различные утилиты получения дампов иногда не могут получить полный дамп. Если у вас возникли проблемы при получении полного дампа, попробуйте получить дамп miniplus, чтобы задействовать возможность восстановления.

При создании дампов miniplus можно использовать другие параметры ProcDump, что продемонстрировано в примерах ниже. Чтобы получить один дамп miniplus приложения store.exe, выполните следующую команду:

```
procdump -mp store.exe
```

Следующая команда позволяет получить один дамп miniplus при сбое в store.exe:

```
procdump -mp -e store.exe
```

Следующая команда получает три дампа miniplus с интервалом 15 секунд:

```
procdump -mp -n 3 -s 15 store.exe
```

Чтобы получить три дампа miniplus, когда счетчик производительности RPC Averaged Latency превышает значение 250 мс в течение 15 секунд, используйте следующую команду:

```
procdump -mp -n 3 -s 15 -p «\MSExchangeIS\RPC Averaged Latency» 250 store.exe
```

Примечание Я не рекомендую получать дампы miniplus для управляемых (.NET) приложений. Создавайте вместо этого полные дампы (-ma). Алгоритм дампов miniplus пытается получить полный дамп для таких приложений, но так он создается на основе минидампа, полученный дамп не является полным. Полный дамп требуется потому, что отладчику нужны полные структуры данных GC и доступ к образу NGEN (чего нельзя получить из хранилища символов или исполняемых файлов).

Неинтерактивный запуск ProcDump

ProcDump не требует запуска в интерактивном сеансе рабочего стола. Неинтерактивный запуск может потребоваться для мониторинга длительно работающего процесса, чтобы не оставаться в системе на все время мониторинга, для решения проблемы, возникающей в отсутствие в системе пользователей или во время выхода из системы.

В следующем примере показано, как использовать PsExec для запуска ProcDump от имени системы в том же неинтерактивном сеансе и на том же рабочем столе, которые используют службы, запущенные от имени системы. Запуск выполняется в экземпляре Cmd.exe, чтобы консольный выход можно было перенаправить в файлы. Обратите внимание на символ escape-последовательности (^) перед символом перенаправления выхода (>), указывающего на то, что последний не должен считаться символом перенаправления выхода PsExec, а является частью командной строки Cmd.exe. Весь пример должен набираться как единая строка команды. (Подробнее о PsExec см. гл. 6; подробнее о неинтерактивных сеансах и рабочих столах см. гл. 2.)

```
psexec -s -d cmd.exe /c procdump.exe -e -t testapp c:\temp\testapp.dmp ^>
c:\temp\procdump.out 2^> c:\temp\procdump.err
```

Если в целевом приложении возникает сбой во время выхода пользователя из системы, такая команда будет работать лучше, чем запуск ProcDump в том же сеансе, поскольку ProcDump может завершить работу раньше, чем целевое приложение. Если же при выходе из системы целевое приложение принудительно завершается, ProcDump не сможет получить дамп. ProcDump действует как отладчик для целевого процесса, а выход из системы отключает все отладчики, подключенные к принудительно завершаемым процессам.

Заметьте также, что ProcDump не может выполнять мониторинг зависших окон приложений, когда целевой процесс работает на другом рабочем столе.

Захват всех сбоев приложений с помощью ProcDump

ProcDump можно использовать и для создания дампа при сбое приложения, назначив эту утилиту «посмертным» отладчиком[1]. В разделе реестра HKLM\Software\Microsoft\Windows NT\CurrentVersion\AeDebug введите в качестве значения «Debugger» REG_SZ командную строку ProcDump, используя %ld как заменитель для PID сбойного процесса. Например, следующая команда будет создавать полный дамп памяти в папке C:\Dumps, когда любое приложение завершается из-за сбоя, используя в качестве имени файла имя процесса и временной штамп:

```
«C:\Program Files\Sysinternals\procdump.exe» /accepteula -ma %ld C:\Dumps
```

Важно указать путь к файлу дампа. В противном случае ProcDump будет пытаться создать дамп в текущей папке, которой при таком способе запуска будет %SystemRoot%\System32. Так как назначенный отладчик запускается в том же контексте безопасности, что и сбойный процесс, ProcDump не сможет создать файл дампа в этой папке, если процесс не имеет административных прав. Заметьте также, что целевая папка должна существовать и в нее должна быть разрешена запись.

---

[1] Это можно сделать и при помощи Windows Error Reporting, но ProcDump легче настроить.

Просмотр дампа в отладчике

Во всех дампах, полученных при выполнении условия, ProcDump записывает комментарии о причине их получения. Такие комментарии можно увидеть в первоначальном тексте, который WinDbg показывает при открытии файла дампа. Первая строка комментария описывает командную строку ProcDump, использованную для создания дампа. Вторая строка комментария описывает причину получения дампа и другие связанные данные, если таковые доступны. Например, если превышен порог памяти, в комментарии будет сказано о пределе выделения памяти и использовании выделения памяти процессом:

```
*** Process exceeded 100 MB commit usage: 107 MB
```

Если превышен порог использования ЦП, в комментарии будет указан порог ЦП, длительность и идентификатор потока (TID), использовавшего наибольшее число циклов ЦП в тот промежуток времени:

```
*** Process exceeded 50% CPU for 3 seconds. Thread consuming CPU: 4484 (0x1184)
```

Если превышен порог счетчика производительности, в комментарии будет указан счетчик производительности, порог, длительность и TID потока, использовавшего наибольшее число циклов ЦП в тот промежуток времени. (Показанный ниже текст должен быть выведен в одной строке с пробелом между точкой и словом «Thread».)

```
*** Counter «\Process(notepad_1376)\% Processor Time»  exceeded  5 for 3
seconds. Thread consuming CPU: 1368 (0x558)
```

Если дамп создан из-за зависшего окна, комментарий будет включать описатель окна в шестнадцатеричном виде. Если дамп был получен незамедлительно, был отложен или инициирован исключением или обычным завершением работы, в комментарии будет указана только причина без дополнительных данных.

Чтобы вам не пришлось изменять контекст занятого потока (команда ~~[TID]s) при открытии дампа, созданного из-за превышения порога ЦП или счетчика производительности, ProcDump вставляет фальшивое исключение, делающее это за вас. Это очень полезно при получении нескольких файлов дампа, так как можно открывать каждый файл, зная, что интересующим вас потоком является контекст потока, выбранный по умолчанию. Вставка фальшивого исключения в дамп приводит к тому, что отладчик сообщает об ошибке следующим текстом:

```
This dump file has an exception of interest stored in it.
The stored exception information can be accessed via .ecxr.
(104c.14c0): Wake debugger - code 80000007 (first/second chance not available)
eax=000cfe00 ebx=00188768 ecx=00000001 edx=00000000 esi=00000000 edi=00000000
eip=01001dc7 esp=00feff70 ebp=00feff88 iopl=0         nv up ei pl zr na pe nc
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b     efl=00000246
```

Теперь вы об этом знаете и можете игнорировать такие сообщения.

DebugView

DebugView — приложение для мониторинга отладочного вывода с локального или удаленного компьютера. В отличие от большинства отладчиков, DebugView может отображать выходные данные пользовательского режима, полученных от всех процессов одного сеанса, а также выходные данные режима ядра. DebugView предоставляет гибкие параметры входа в систему и отображения данных и работает во всех версиях (x86 и x64) Windows XP и последующих ОС.

Отладочный вывод

В Windows есть API, позволяющие программам отправлять текст, который может быть получен и отображен отладчиком. Если нет активного отладчика, эти API ничего не делают. Такие интерфейсы облегчают предоставление диагностических данных, которые могут быть получены любым стандартным отладчиком или проигнорированы, если отладчик не подключен.

Отладочный вывод может генерироваться как программами пользовательского режима, так и драйверами режима ядра. Для программ пользовательского режима Windows предоставляет Win32 API *OutputDebugString*. 16-разрядные приложения, работающие на платформе x86, могут предоставлять выходные данные отладки вызовом Win16 API *OutputDebugString*, который перенаправляется интерфейсу Win32 API. Для управляемых приложений в .NET Framework имеются классы *System.Diagnostics.Debug* и *Trace* со статическими методами, выполняющими внутренние вызовы *OutputDebugString*. Эти методы можно вызывать также из Windows PowerShell:

```
[System.Diagnostics.Debug]::Print(«Выходные данные отладки»)
```

Драйверы режима ядра могут генерировать отладочный вывод, вызывая подпрограммы *DbgPrint* и *DbgPrintEx* или соответствующие функции. Программисты могут также использовать макросы *KdPrint* и *KdPrintEx*, генерирующие отладочный вывод только в отладочных сборках и “молчащие” в итоговых сборках.

Windows предоставляет две реализации API *OutputDebugString*, — ANSI и Unicode, — но при внутренней обработке выходные данные кодируются в ANSI. Unicode-реализация преобразует текст отладки, используя параметры локальной системы, и передает его реализации ANSI. В результате, некоторые символы Unicode могут отображаться некорректно.

Экран DebugView

Запустите программу DebugView (Dbgview.exe). Она немедленно начнет перехват и отображение вывода отладки Win32, полученных со всех рабочих столов текущего сеанса сервера терминалов.

Примечание Все интерактивные сеансы рабочего стола реализованы как сеансы сервера терминалов.

Как видно из рис. 7-13, первый столбец содержит порядковый номер, назначенный программой. В порядковых номерах могут появляться пропуски, когда правила фильтрации исключают строки текста или внутренние буферы DebugView переполняются при очень активной работе. Порядковые номера сбрасываются каждый раз, когда экран очищается. (О фильтрации рассказывается далее в этой главе.)

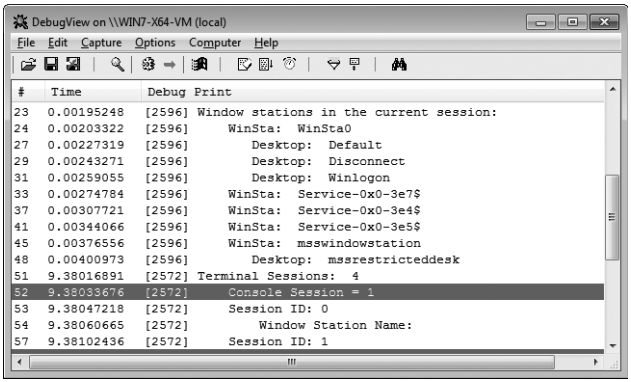


Рис. 7-13. DebugView

Во втором столбце указано время захвата элемента, выраженное в прошедшем времени или времени системных часов. По умолчанию отображается число секунд, прошедшее с момента получения первой отладочной записи, и первый элемент всегда имеет время 0.00, что может быть полезным при отладке проблем, связанных со временем. Данный таймер сбрасывается при очистке экрана. Если хотите, чтобы вместо этого отображалось локальное время, выберите в меню **Options (Параметры)** команду **Clock Time (Системное время)**. Если требуется, чтобы временной штамп содержал миллисекунды, выберите команду меню **Options | Show Milliseconds (Параметры | Показывать миллисекунды)**. Настраивать отображение времени можно также с помощью параметров командной строки `/o` (для отображения системного времени), `/om` (для отображения системного времени с миллисекундами) и `/on` (для отображения прошедшего времени).

Совет При включении отображения миллисекунд параметры уже показанных записей не меняются. Можно обновить эти записи, дважды нажав Ctrl+T, чтобы отключить и снова включить отображение системного времени. После этого все записи будут показывать миллисекунды.

Выходные данные отладки отображаются в столбце **Debug Print (Печать отладки)**. Выходные данные пользовательского режима содержат идентификатор процесса (PID), сгенерировавшего выходные данные, заключенный в квадратные скобки, за которым идут сами данные. Если отображение PID не нужно, отключите параметр **Win32 PIDs** в меню **Options (Параметры)**.

Можно выбрать одну или несколько строк выходных данных отладки и скопировать их в буфер обмена, нажав Ctrl+C. DebugView поддержива-

ет стандартные способы выделения нескольких строк с Shift и клавишами-стрелками вверх и вниз (для выделения смежных строк) и щелчки с Ctrl (для выделения несмежных строк).

По умолчанию включен параметр **Force Carriage Returns (Принудительный возврат каретки)**, чтобы каждая строка отладочного вывода появлялась на новой строке, независимо от того, завершается ли переданный текст символом возврата каретки. Если отключить этот параметр в меню **Options (Параметры)**, DebugView будет помещать текст в буфер и выводить на экран только в том случае, если встречается символ возврата каретки или если буфер заполняется (его размер — примерно 4192 символа). Это позволяет приложениям и драйверам создавать строки, содержащие несколько вызовов функций, генерирующих отладочный вывод. Но если выходные данные генерируются несколькими процессами, они могут смешиваться, и PID, указанный в строке, будет идентификатором процесса, передавшего символ возврата каретки или заполнившего буфер.

Если текст не умещается в столбце, наведите на него указатель мыши, полый текст откроется во всплывающей подсказке.

Новые выходные данные отладки добавляются в конец списка. Функция **Autoscroll (Автопрокрутка)**, отключенная по умолчанию, прокручивает экран при появлении новых записей, чтобы были видны последние данные. Чтобы включить или отключить эту функцию, нажмите Ctrl+A или щелкните на значке **Autoscroll (Автопрокрутка)** на панели инструментов.

Выходные данные можно комментировать, выбрав команду **Append Comment (Добавить комментарий)** в меню **Edit (Правка)**. Текст, введенный в диалоговом окне **Append Comment (Добавление комментария)**, будет добавлен к выходным данным на экране и в файл журнала, если ведение журнала включено. Заметьте, что правила фильтрации применяются к добавленным комментариям так же, как и к отладочному выводу.

Можно увеличить место для отображения вывода, выбрав в меню **Options (Параметры)** команду **Hide Toolbar (Скрыть панель инструментов)**. Можно также увеличить число видимых строк, выбрав меньший размер шрифта. Чтобы изменить шрифт, выберите в меню **Options (Параметры)** команду **Font (Шрифт)**.

Чтобы DebugView запустился в фоновом режиме, не занимая места на панели задач, выберите в меню **Options (Параметры)** команду **Hide When Minimized (Скрывать при сворачивании)**. Если после этого свернуть окно DebugView, его будет видно только как значок в области уведомлений (также известной как «трей»). Можно щелкнуть на этом значке правой кнопкой мыши, чтобы открыть контекстное меню **Capture (Захват)**, с помощью которого можно включать и отключать различные параметры захвата. Чтобы снова открыть окно программы, дважды щелкните на этом значке. Параметр **Hide When Minimized** можно включить при запуске, добавив **/t** в командную строку.

Если выбрать в меню **Options (Параметры)** команду **Always On Top (Всегда сверху)**, окно DebugView всегда будет отображаться поверх других окон на рабочем столе, если его не свернуть.

Получение отладочного вывода пользовательского режима

DebugView может получать отладочный вывод из нескольких локальных источников: текущий сеанс служб терминалов, глобальный сеанс служб терминалов («сеанс 0») и режим ядра. Каждый из этих источников можно выбрать в меню **Capture (Захват)**. Весь захват можно включить или отключить, выбрав команду **Capture Events (Захват событий)**, нажав Ctrl+E или щелкнув на значке **Capture (Захват)** на панели инструментов. Если захват событий отключен, вывод не принимается, а если включен, данные принимаются из выбранных источников.

По умолчанию DebugView получает выходные данные только от текущего сеанса служб терминалов, которому соответствует команда **Capture Win32 (Захват Win32)** меню **Capture (Захват)**. Сеанс служб терминалов представляет всю активность пользовательского режима, связанную с интерактивным рабочим столом. Сюда входят все процессы, работающие в оконных станциях и рабочих столах (Win32) этого сеанса.

В Windows XP и Windows Server 2003 интерактивный сеанс может находиться в сеансе 0 и находится там всегда, когда не задействовано быстрое переключение пользователей и удаленный рабочий стол. Кроме того, в сеансе 0 выполняются все службы и определяются *глобальные* объекты. Если DebugView выполняется в сеансе 0, и включен захват Win32 (**Capture Win32**), будут захватываться выходные данные отладки от служб и процессов интерактивного пользователя. Для получения выходных данных отладки из текущего сеанса, даже если это данные служб, администраторские права не требуются (подробнее см. в главе 2.)

Когда включено быстрое переключение пользователей или удаленный рабочий стол, пользователи Windows XP и Windows Server 2003 часто входят в сеансы, отличные от глобального. Кроме того, начиная с Windows Vista, изоляция сеанса 0 гарантирует, что пользователи никогда не входят в тот сеанс, в котором работают службы. При работе не в нулевом сеансе DebugView добавляет в меню **Capture (Захват)** команду **Capture Global Win32 (Захват глобальных данных Win32)**. Если включить этот параметр, будут захватываться выходные данные от процессов, работающих в сеансе 0. Для использования этого параметра в Windows Vista и последующих версиях ОС, DebugView необходимы повышенные привилегии. Для включения этого параметра в Windows XP администраторские права не требуются.

Получение отладочного вывода режима ядра

DebugView можно настроить для получения выходных данных отладки режима ядра, генерируемых драйверами устройств и ядром Windows, включив

параметр **Capture Kernel (Захват ядра)** в меню **Capture (Захват)**. В выходных данных режима ядра не указываются идентификаторы процессов, так как эти данные обычно не относятся к контексту процессов. Для получения данных режима ядра требуются администраторские права, в частности, привилегия Load Driver (Загрузка драйверов).

Компоненты режима ядра могут указывать уровень серьезности каждого отладочного сообщения. В Windows Vista и последующих версиях выходные данные режима ядра можно фильтровать по уровню серьезности. Если хотите получать все выходные данные ядра, выберите в меню **Capture (Захват)** команду **Enable Verbose Kernel Output (Включить подробные выходные данные ядра)**. Если этот параметр не включен, DebugView захватывает только выходные данные с уровнем серьезности «ошибка».

Можно настроить DebugView так, чтобы выходные данные отладки режима ядра передавались отладчику режима ядра или поглощались. Переключать режим передачи можно при помощи меню **Capture (Захват)** или значка **Pass-Through (Передача)** на панели инструментов. Режим передачи позволяет просматривать выходные данные режима ядра в выходных буферах стандартного отладчика режима ядра и в DebugView одновременно.

Так как DebugView — интерактивная программа, ее нельзя запустить до входа в систему. Обычно для просмотра выходных данных отладки, сгенерированных до входа в систему, требуется подключать отладчик ядра с удаленного компьютера. Функция DebugView Log Boot предоставляет альтернативу, захватывает выходные данные режима ядра во время запуска системы, держа эти данные в памяти и отображая их после входа в систему и интерактивного запуска DebugView. Если выбрать в меню **Capture (Захват)** команду **Log Boot (Запись загрузки)**, DebugView настроит свой драйвер ядра так, чтобы он загружался в начале следующей последовательности загрузки. Когда он загружается, он создает буфер размером 4 МБ, в который записывает подробные выходные данные отладки режима ядра, пока буфер не заполняется или пока DebugView не подключается к нему. Если запустить DebugView с администраторскими правами и включенной функцией **Capture Kernel (Захват в режиме ядра)**, DebugView проверит существование такого буфера в памяти ядра. Если буфер найден, DebugView отображает его содержимое. Настройка журнала загрузки требует администраторских разрешений и применяется только при следующей загрузке.

Если выходные данные отладки режима ядра получаются во время проверки ошибок (также известной как сбой с синим экраном), DebugView может восстановить выходные данные, полученные к тому моменту из файла дампа сбоя. Это может быть полезным, например, при попытках диагностировать сбой с участием разрабатываемого вами драйвера режима ядра. Вы можете также оснастить свой драйвер средствами для записи отладочного вывода, чтобы пользователи, у которых произошел сбой этого драйвера, могли отправить вам файл с выходными данными отладки, а не дамп памяти целиком.

Чтобы открыть файл дампа краха для анализа в DebugView, выберите в меню **File (Файл)** команду **Process Crash Dump (Обработать дамп краха)**. DebugView выполнит в файле поиск буферов выходных данных отладки. Если они найдены, DebugView предложит ввести имя файла журнала, в котором будут сохранены выходные данные. Сохраненные файлы выходных данных можно загружать в DebugView для просмотра. Заметьте, что система должна быть настроена для создания дампа ядра или полного дампа (а не минидампа), иначе эта функция не будет работать. При выходе DebugView сохраняет все параметры настройки захвата и восстанавливает их при следующем запуске. Если программа была запущена с повышенными правами и получала выходные данные ядра или сеанса 0, то при следующем запуске от имени того же пользователя, но без администраторских прав, она покажет сообщение об ошибке и отключит эти параметры, так как не сможет получать данные из этих источников. Избежать появления таких ошибок можно, запуская DebugView с параметром `/kn`, отключающим захват данных ядра, и параметром `/gn`, отключающим захват глобальных данных.

Поиск, фильтрация и выделение выходных данных

DebugView имеет несколько функций, помогающих сосредоточиться на нужных фрагментах выходных данных: поиск, фильтрация, выделение и ограничение числа строк, отображаемых на экране.

Очистка экрана

Чтобы очистить экран от всего полученного отладочного текста, нажмите Ctrl+X или щелкните на значке **Clear (Очистить)** на панели инструментов. Очистку экрана может инициировать источник выходных данных отладки: когда в любом месте входной строки встречается текст DBGVIEWCLEAR (все буквы заглавные), DebugView очищает экран. При очистке выходных данных порядковый номер и таймер прошедшего времени сбрасываются на 0.

Поиск

Если нужно найти строку, содержащую определенный текст, нажмите Ctrl+F, чтобы открыть окно **Find (Найти)**. Если указанный текст совпадает с текстом в окне с выходными данными, DebugView выделяет следующую подходящую строку и отключает функцию автопрокрутки, чтобы эта строка отображалась в окне. Чтобы повторить поиск, нажмите F3. Чтобы изменить направление поиска, нажмите Shift+F3.

Фильтрация

Еще один способ изоляции нужных выходных данных — фильтрация. Щелкните кнопку **Filter/Highlight (Фильтровать/Выделить)** на панели инструментов DebugView, чтобы открыть окно **Filter (Фильтр)** (рис. 7-14). Поля **Include (Включить)** и **Exclude (Исключить)** используются для указания условий для включения и исключения входящих строк отладочного текста на основе их содержимого. В секции **Highlight (Выделение)** можно вы-

брать цвет для выделения строк в зависимости от их содержимого. Условия фильтрации и выделения можно сохранить на диске и повторно загружать в дальнейшем (выделение обсуждается в следующем разделе.)

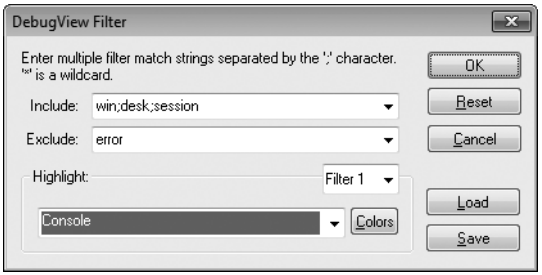


Рис. 7-14. Окно DebugView Filter

В поле **Include** введите текст, содержащийся в строках выходных данных, которые вы хотите отображать в окне DebugView, а в поле **Exclude** — текст, строки с которым отображаться не должны. Можно ввести несколько выражений, разделив их точкой с запятой. Не включайте пробелы в выражения фильтра, если не хотите, чтобы они стали частью фильтра. Заметьте, что символ «*» интерпретируется как знак подстановки, а выражения нечувствительны к регистру и применяются также к части стоки, содержащей идентификатор процесса, если идентификатор включен в выходные данные. По умолчанию включено все («*»), и ничего не исключено.

Если вы хотите включать в выходные данные строки, содержащие текст «win», «desk» и «session», если только они не содержат также текст «error», введите в фильтр **Include** текст «win;desk;session» (без кавычек), а в фильтр **Exclude** — текст «error» (как на рис. 7-14). Если вы хотите, чтобы в DebugView отображались только данные, имеющие текст «MyApp:» и слово «severe», стоящее далее в строке, используйте знак подстановки в фильтре включения: «myapp:*severe».

Фильтрация применяется только к строкам и комментариям, добавленным после установки фильтра. Новые строки текста, удовлетворяющие условиям фильтрации, отображаются, а не удовлетворяющие строки отбрасываются, их нельзя будет вернуть, изменив правила фильтрации. Кроме того, изменение правил фильтрации не удаляет строки, уже отображающиеся в DebugView.

Если при выходе из DebugView действовали какие-то правила фильтрации, DebugView покажет их в диалоговом окне при следующем запуске. Просто щелкните OK, чтобы продолжить использовать их, или укажите новые правила. Можно изменить их вручную, загрузить ранее сохраненный фильтр, щелкнув кнопку Load (Загрузить), или удалить фильтр, щелкнув кнопку Reset (Сбросить). Можно пропустить это окно и продолжить использовать предыдущие правила, добавив в командную строку DebugView параметр /f.

Выделение

Выделение позволяет изменять цвет строк в зависимости от их текста. DebugView поддерживает до 20 отдельных правил выделения, каждое из которых имеет собственные цвета фона и текста. Синтаксис правил выделения такой же, как для фильтра включения.

Раскрывающийся список **Filter (Фильтр)** из группы **Highlight (Выделение)** позволяет выбрать фильтр (с номером от 1 до 20), который вы хотите изменить. По умолчанию каждый фильтр связан с комбинацией цветов, но не с правилом выделения. Чтобы установить правило для этого фильтра, введите текст правила в раскрывающийся список, показывающий комбинацию цветов. На рис. 7-14 первый фильтр **(Filter 1)** выделяет строки, содержащие слово «Console».

Фильтры с меньшими номерами имеют преимущество перед фильтрами с большими номерами. Если строка текста подходит под правила фильтра 3 и фильтра 5, она будет выделена цветами фильтра 3. При изменении правил выделения все строки на экране обновляются в соответствии с ними.

Чтобы изменить цвета, связанные с фильтром выделения, выберите фильтр в раскрывающемся списке и щелкните кнопку **Colors (Цвета)**. Чтобы изменить цвет переднего плана, установите переключатель **FG**, выберите цвет и щелкните кнопку **Select (Выбрать)**. Сделайте то же самое, установив переключатель **BG**, чтобы изменить цвет фона, и щелкните **OK**.

Сохранение и восстановление правил фильтрации и выделения

Кнопки **Load (Загрузить)** и **Save (Сохранить)** окна **Filter (Фильтр)** позволяют сохранять и восстанавливать параметр фильтрации, включая правила включения, исключения и выделения, а также цвета выделения. DebugView использует для своих файлов параметров фильтрации расширение .INI, хотя формат этих файлов отличается от формата файлов инициализации.

Кнопка **Reset (Сброс)** сбрасывает все правила фильтрации и выделения, возвращая параметры по умолчанию. Обратите внимание на то, что эта кнопка не восстанавливает стандартные цвета выделения.

Период журнала

Еще один способ настройки выходных данных DebugView — ограничение числа строк, отображаемых на экране. Выберите в меню **Edit (Правка)** команду **History Depth (Период журнала)**, чтобы открыть одноименное окно. Введите число, и в окне DebugView будет сохраняться именно столько строк, а остальные будут удаляться. Значение 0 (ноль) отключает ограничение на число строк. Период журнала можно указать в командной строке, указав параметр `/h` и число строк.

Нет необходимости использовать эту функцию для предотвращения заполнения всей виртуальной памяти системы при длительном мониторинге: DebugView следит за использованием системной памяти, предупреждает пользователя и приостанавливает получение выходных данных отладки, когда остается мало памяти.

Сохранение, запись в журнал и печать

DebugView позволяет сохранять полученный отладочный вывод в файл при получении или по требованию. Сохраненные файлы можно открывать и отображать в DebugView, а также выводить на печать целиком или частично.

Можно сохранить содержимое окна DebigView в текстовом файле, выбрав в меню **File (Файл)** команду **Save (Сохранить)** или **Save As (Сохранить как)**. По умолчанию используется расширение .LOG. Формат файлов — текст ANSI, разделенный знаками табуляции. Чтобы открыть сохраненный текст, выберите команду меню **File | Open (Файл | Открыть)** или укажите путь к файлу в командной строке:

```
dbgview c:\temp\wi n7-x86-vm.log
```

Запись в журнал

Чтобы выходные данные записывались в файл журнала сразу, при отображении, выберите в меню **File (Файл)** команду **Log To File (Запись в файл)**. При первом выборе этой команды или нажатии кнопки **Log To File** на панели инструментов DebugView открывает окно **Log-To-File Settings (Параметры записи в файл)**, предлагающее указать путь к файлу (рис. 7-15). После этого команда **Log To File** и одноименная кнопка только включают и отключают запись в журнал. Чтобы сменить файл или изменить другие параметры записи, выберите в меню **File (Файл)** команду **Log To File As (Запись в формате...)**. (Если в текущий момент ведется запись в файл, выполнение этой команды приведет к отключению записи.)

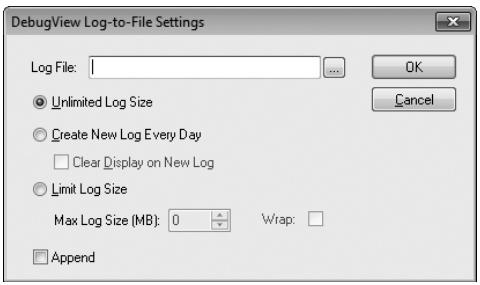


Рис. 7-15. Окно DebugView Log-to-File Settings

Перечислим остальные параметры настройки, доступные в диалоговом окне **Log To File**:

- **Unlimited Log Size (Неограниченный размер журнала)** — позволяет файлу журнала расти без ограничений;
- **Create New Log Every Day (Создавать новый журнал каждый день)** — размер файла журнала не ограничивается, но каждый день создается новый файл с текущей датой в имени. Можно также включить очистку экрана при создании нового файла;
- **Limit Log Size (Ограничить размер журнала)** — файл журнала не увеличивается сверх указанного размера. Запись будет остановлена, если не

установить флажок **Wrap (Перенос)**. Если этот параметр включен, запись будет вестись в начало файла.

Если флажок **Append (Добавлять)** не установлен, и целевой файл уже существует, DebugView очистит этот файл, когда начнет запись в него. Если флажок установлен, новые данные будут добавляться в конец файла.

Если мониторинг выходных данных отладки выполняется с нескольких удаленных компьютеров, и включена запись в файл, все выходные данные записываются в один файл. Фрагменты данных от разных компьютеров разделяются заголовком, указывающим имя компьютера, от которого получены последующие записи.

Параметры записи в журнал можно контролировать также с помощью параметров командной строки (табл. 7-2).

Табл. 7-2. Параметры командной строки для записи в журнал

Параметр	Описание
-l *файл_журнала*	Записывает выходные данные в указанный файл
-m *n*	Ограничивает размер файла журнала до *n* МБ
-p	Добавляет записи в конец файла, если он существует
-w	Используется вместе с –m; выполняет переход в начало файла при достижении максимального размера
-n	Создает новый файл журнала каждый день, указывая текущую дату в имени файла
-x	Используется вместе с –n; очищает экран при создании нового файла журнала

Печать

Чтобы распечатать содержимое экрана на принтере, выберите в меню **File (Файл)** команду **Print (Печать)** или **Print Range (Печать диапазона)**. Первая команда выводит на печать все записи выходных данных, вторая — только записи с порядковыми номерами из указанного диапазона. Прежде чем распечатать данные, захват необходимо отключить.

В диалоговом окне **Print Range (Печать диапазона)** можно также указать, должны ли вместе с данными печататься порядковые номера и временные отметки. Если эти поля не нужны, их можно не печатать, чтобы сэкономить бумагу. Установленные параметры используются во всех последующих операциях печати.

Чтобы предотвратить перенос строк, ширина которых больше ширины листа, можно использовать альбомную ориентацию при печати.

Удаленный мониторинг

DebugView поддерживает удаленный мониторинг, позволяя просматривать отладочный вывод, сгенерированный на удаленных системах. DebugView может подключаться к нескольким удаленным компьютерам и локальному

компьютеру одновременно. Можно переключать режимы просмотра, чтобы просматривать данные с определенного компьютера, выбирая его в меню **Computer (Компьютер)** (рис. 7-16), или переключаться между компьютерами, нажимая Ctrl+Tab. Активный компьютер указан на панели заголовка и отмечен стрелкой в меню **Computer**. Можно также открыть каждый компьютер в отдельном окне и просматривать все данные одновременно.

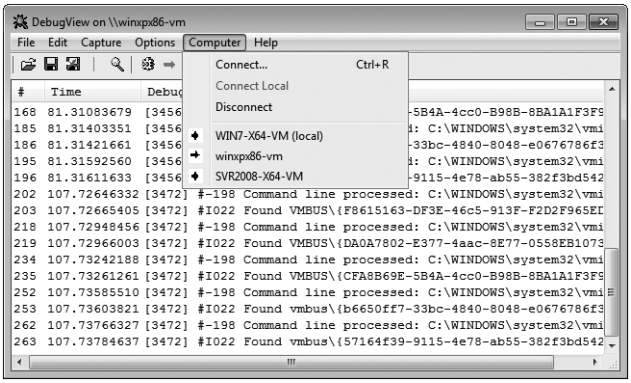


Рис. 7-16. Одновременный мониторинг локального компьютера и двух удаленных

Для удаленного мониторинга DebugView запускается на удаленной системе в режиме агента, отправляя полученный вывод серверному экземпляру DebugView, который отображает их. Обычно запуск DebugView в режиме агента на удаленном компьютере производится вручную. В некоторых случаях центральный экземпляр может автоматически установить и запустить удаленный агент, но при использовании брандмауэров это обычно невыполнимо. Чтобы начать удаленный мониторинг, нажмите Ctrl+R или выберите в меню **Computer** команду **Connect (Подключить)**, чтобы открыть окно подключения. Введите имя или IP-адрес удаленного компьютера или выберите из раскрывающегося списка компьютер, к которому подключались ранее, и щелкните **OK**. DebugView попытается установить и запустить агент на этом компьютере; если это не удастся, будет произведена попытка поиска и подключения к уже запущенному агенту. Если это удалось, DebugView начнет отображение выходных данных отладки, полученных от этого компьютера, добавив его имя в панель заголовка и меню **Computer**.

Чтобы начать мониторинг локального компьютера, выберите в меню **Computer** команду **Connect Local (Подключиться локально)**. Будьте внимательны, чтобы не подключить несколько экземпляров DebugView к одному компьютеру, так как в этом случае каждый экземпляр будет получать только часть данных.

Чтобы одновременно просматривать выходные данные от двух компьютеров, выберите в меню **File (Файл)** команду **New Window (Новое окно)**, чтобы открыть новое окно DebugView, перед установкой второго подключения. Устанавливайте второе подключение из нового окна.

Чтобы остановить мониторинг удаленного компьютера, сделайте его активным, выбрав его в меню **Computer**, а затем выберите команду **Disconnect (Отключить)** из меню **Computer**.

Запуск агента DebugView

Чтобы вручную запустить DebugView в режиме агента, добавьте в командную строку аргумент /a. Пока выполняется запуск, будет отображаться окно Waiting for connection (Ожидание подключения) (рис. 7-17). Затем в этом окне появится сообщение Connected (Подключение выполнено). Заметьте, что в режиме агента DebugView не захватывает и не сохраняет никакие данные, если нет подключения к монитору DebugView. При подключении агент всегда захватывает выходные данные Win32 из текущего сеанса служб терминалов. Чтобы агент захватывал выходные данные отладки ядра, добавьте в командную строку параметр /k; чтобы получать подробные выходные данные ядра, добавьте в командную строку параметр /v. Чтобы получать глобальные выходные данные (из сеанса 0), добавьте в командную строку параметр /g.

Рис. 7-17. Окно агента DebugView

Если монитор отключается или подключение прерывается по другим причинам, в окне состояния агента снова появляется сообщение Waiting for connection (Ожидание подключения), и DebugView ожидает нового подключения. Чтобы получать сообщение о возникающей ошибке, добавьте в командную строку агента параметр /e. Новое подключение не будет установлено, пока пользователь не закроет окно сообщения.

Можно скрыть окно состояния агента, добавив в командную строку параметр /t; вместо окна будет отображаться значок в области уведомлений панели задач. Когда агент не подключен к монитору, этот значок будет серым, и цветным, когда подключение установлено. Чтобы открыть окно состояния, дважды щелкните на этом значке. При сворачивании окно снова исчезнет, оставив только значок. Можно совсем скрыть пользовательский интерфейс агента, добавив параметр /s. При этом агент DebugView останется активным до тех пор, пока пользователь не выйдет из системы, и будет принимать подключения мониторов в тихом режиме. Параметр /s перекрывает параметр /e: если монитор отключается, агент будет ожидать в тихом режиме и принимать новые подключения без отображения уведомлений.

Агент, запущенный вручную, прослушивает подключения к порту TCP 2020. Брандмауэр Windows может показать предупреждение при первом запуске DebugView в режиме агента. Если пользователь разрешит доступ, указанный в предупреждении, Windows создаст исключение брандмауэра для

DebugView, без него запущенный вручную агент не будет работать. Заметьте, что подключения устанавливаются анонимно, без аутентификации.

Если монитор реализован как служба Windows, агент автоматически устанавливается и запускается на удаленном компьютере. Следовательно, он работает в сеансе 0 служб терминалов, где может выполнять мониторинг только ядра и глобального отладочного вывода Win32 и не может получать выходные данные от сеансов интерактивных пользователей, работающих вне сеанса 0. Кроме того, он прослушивает подключения к произвольному порту с большим номером, которые нельзя установить при использовании брандмауэра. В большинстве случаев намного надежнее будет запустить агент вручную.

При использовании агента, автоматически установленного монитором, состояние глобального захвата, захвата отладочных данных Win32, захвата данных режима ядра и новых удаленных сеансов будет определяться текущими параметрами монитора DebugView. Изменения в этих параметрах будут применяться к наблюдаемому компьютеру незамедлительно.

LiveKd

LiveKd позволяет использовать отладчики ядра для просмотра моментального снимка работающей системы, не перезагружая систему в режиме отладки. Это полезно при решении проблем, возникающих в режиме ядра на компьютере, не загруженном в режиме отладки. Некоторые проблемы плохо воспроизводятся, поэтому перезагрузка системы может помешать. Кроме того, загрузка компьютера в режиме отладки изменяет поведение некоторых подсистем, что может еще больше затруднить анализ. LiveKd не только не требует загрузки в режиме отладки, но и позволяет отладчикам Microsoft выполнять некоторые действия, доступные только при локальной отладке ядра, такие как создание полного дампа памяти.

В дополнение к проверке локальной системы, LiveKd поддерживает внешнюю отладку виртуальных машин Hyper-V с хоста Hyper-V. В этом режиме отладчик запускается на хосте Hyper-V, а не в виртуальной машине, поэтому не нужно копировать какие-либо файлы на целевую виртуальную машину или настраивать ее.

LiveKd создает файл дампа с моментальным снимком памяти ядра, не останавливая ядро во время его создания. Затем LiveKd предоставляет этот файл выбранному отладчику ядра. После этого можно использовать отладчик для выполнения любых операций с этим снимком памяти работающего ядра, как с обычным файлом дампа.

Поскольку LiveKd получает дамп физической памяти, возможны ситуации, когда структуры данных находятся в процессе изменения системой и не являются согласованными. Запускаясь, отладчик каждый раз начинает работать со свежим представлением состояния системы. Если вы хотите обновить моментальный снимок, выйдите из отладчика (команда q), и LiveKd

спросит, хотите ли вы запустить его снова. Если отладчик зацикливается при выводе, нажмите Ctrl+C, чтобы прервать операцию, и перезапустите его. Если он завис, нажмите Ctrl+Break; процесс отладчика будет завершен, и система спросит, нужно ли запустить отладчик снова.

Требования LiveKd

LiveKd поддерживает все версии Windows x86 и x64. Запускать эту утилиту необходимо с администраторскими правами, включая привилегию Debug (Отладка).

Работа LiveKd зависит от утилит отладки для Windows (Debugging Tools for Windows), которые должны быть предварительно установлены на том компьютере, где будет запускаться LiveKd. Скачать эти утилиты можно по адресу *http://www.microsoft.com/whdc/ devtools/debugging/default.mspx*. Раньше установщик утилит отладки можно было скачать отдельно, но теперь они входят в состав SDK Windows. Чтобы установить их, нужно запустить установщик SDK и выбрать установку утилит отладки. Можно также использовать варианты Debugging Tools Redistibutable, которые представлены отдельными установщиками, доступными для архитектур x86, x64 и IA64. Они удобны, если вы хотите установить утилиты отладки на другие компьютеры без установки SDK.

LiveKd требует файлы символов ядра. При необходимости их можно скачать с открытого сервера символов Microsoft. Если анализируемая система не подключена к Интернету, необходимые файлы символов можно взять из другой системы (см. врезку ниже).

Запуск LiveKd

LiveKd имеет следующий синтаксис командной строки:

```
livekd [-w | -k путь_к_отладчику | -o файл_дампа] [[-hvl] | [-hv имя_VM]
[-p]] [параметры отладчика]
```

В табл. 7-3 вкратце описаны параметры командной строки LiveKd. Ниже они рассматриваются подробнее.

Табл. 7-3. Параметры командной строки LiveKd

Параметр	Описание
-w	Запускает WinDbg.exe вместо Kd.exe
-k *путь_к_отладчику*	Запускает указанный отладчик вместо Kd.exe
-o *файл_дампа*	Сохраняет дамп ядра в *файл_дампа* вместо запуска отладчика
-hvl	Используется на хосте Hyper-V; показывает список GUID и имен доступных гостевых виртуальных машин
-hv *имя_VM*	Используется на хосте Hyper-V; выполняет отладку виртуальной машины, указанной по GUID или имени

Табл. 7-3. (*окончание*)

Параметр	Описание
-p	Используется на хосте Hyper-V; приостанавливает целевую виртуальную машину во время получения дампа (рекомендуется использовать вместе с –o)
параметры отладчика	Дополнительные параметры командной строки, передаваемые отладчику ядра

По умолчанию LiveKd получает моментальный снимок локального компьютера и запускает Kd.exe. Параметры –w и –k позволяют указать WinDbg.exe или любой другой отладчик вместо Kd.exe. LiveKd передает отладчику любые дополнительные параметры командной строки, за которыми идет параметр –z и путь к эмулированному файлу дампа.

Чтобы выполнить отладку виртуальной машины Hyper-V с хоста, укажите параметр –hv и имя или GUID этой виртуальной машины. Чтобы получить список GUID и имен доступных виртуальных машин, запустите LiveKd с параметром –hvl. Заметьте, что на хосте можно выполнять отладку только одной виртуальной машины за раз.

При запуске с параметром –o LiveKd просто сохраняет дамп ядра целевой системы в указанный файл и не запускает отладчик. Это полезно для получения дампов системы для анализа в автономном режиме. Если целевой системой является виртуальная машина Hyper-V, можно добавить параметр –p для ее приостановки на время получения моментального снимка, чтобы в дампе не было нарушенных структур данных.

Если запустить отладчик, не указав параметр –k и путь к отладчику, LiveKd выполнит поиск Kd.exe или WinDbg.exe в следующих местах:

- текущая папка, из которой произведен запуск LiveKd;
- папка, в которой находится LiveKd;
- стандартный путь установки утилит Debugging Tools («%ProgramFiles%\Debugging Tools for Windows (x86)» в архитектуре x86 и «%ProgramFiles%\Debugging Tools for Windows (x64)» в архитектуре x64);
- папка, указанная переменной PATH.

Если переменная окружения _NT_SYMBOL_PATH не настроена, LiveKd спросит, нужно ли настроить систему для использования сервера символов Microsoft, а затем запросит локальную папку, в которую будут загружены файлы символов (по умолчанию C:\Symbols).

Информацию об использовании отладчиков ядра см. в документации по утилитам Debugging Tools.

Примечание Отладчик сообщит, что не может найти символы для LiveKdD.SYS. Это нормально, так как я не публиковал символы для LiveKdD.SYS. Отсутствие этих символов не повлияет на работу отладчика.

Примеры использования LiveKD

Следующая команда выполняет отладку моментального снимка локальной системы, передавая WinDbg параметры для записи файла журнала и пропуска диалогового окна сохранения рабочего пространства:

```
livekd -w -Q -logo C:\dbg.txt
```

Следующая команда получает дамп ядра локальной системы и не запускает отладчик:

```
livekd -o C:\snapshot.dmp
```

При выполнении на хосте Hyper-V эта команда возвращает список виртуальных машин, доступных для отладки, а затем показывает образец выходных данных:

```
C:\>livekd  -hvl

Listing active Hyper-V partitions...

Hyper-V VM GUID                          Partition ID    VM Name
---------------------------------------  ------------    ------
3187CB6B-1C8B-4968-A501-C8C22468AB77              29    WinXP x86 (SP3)
9A489D58-E69A-48BF-8747-149344164B76              30    Win7 Ultimate x86
DFA26971-62D7-4190-9ED0-61D1B910466B              28    Win7 Ultimate x64
```

Полученные GUID и имена можно использовать, чтобы указать виртуальную машину для отладки. Следующая команда приостанавливает виртуальную машину «Win7 Ultimate x64» из предыдущего примера и получает дамп ядра этой системы, возобновляя ее работу после получения дампа:

```
livekd -p -o C:\snapshot.dmp -hv DFA26971-62D7-4190-9ED0-61D1B910466B
```

Наконец, следующая команда выполняет отладку моментального снимка «WinXP x86 (SP3)» с использованием Kd.exe:

```
livekd -hv «WinXP x86 (SP3)»
```

Использование LiveKd для получения дампа памяти ядра онлайн Сталкивались ли вы с ситуацией, когда требуется получить дамп памяти ядра, но ваш клиент отказывается подключить целевую систему к Интернету, что не дает загрузить необходимые файлы символов? Я слишком часто имел сомнительное удовольствие разбираться с такой проблемой, поэтому решил описать ее решение здесь.

Основная проблема в том, что нужны файлы символов для дампа памяти ядра. Как минимум, нужно иметь символы для Ntoskrnl.exe. Недостаточно просто загрузить символы для целевой ОС и пакетов обновлений с WHDC или MSDN, поскольку возможны изменения соответствующих символов свежими обновлениями, вышедшими после пакета обновлений.

В этом случае я делаю следующее:

- копирую Ntoskrnl.exe и остальные файлы, для которых нужны символы, из папки System32 компьютера, отладку которого требуется выполнить, и помещаю их в папку (например, C:\DebugFiles) на компьютере, подключенном к Интернету;
- устанавливаю на компьютер, подключенный к Интернету, утилиты Debugging Tools for Windows;
- из командной строки этой системы выполняю команду `Symchk` для загрузки символов для выбранных файлов. Команда будет выглядеть примерно так:

```
symchk /if C:\DebugFiles\*.* /s srv*C:\DebugSymbols*http://msdl.
microsoft, com/download/symbols
```

- копирую загруженные символы на целевую систему (из папки C:\DebugSymbols, если следовать моему примеру);
- устанавливаю утилиты Debugging Tools for Windows на том компьютере, с которого требуется получить дамп памяти ядра, и копирую LiveKd.exe в ту же папку вместе с отладчиками. Добавляю эту папку в переменную PATH;
- открываю командную строку с администраторскими привилегиями и записываю в переменную окружения _NT_SYMBOL_PATH путь к папке с файлами символов. Например:

```
SET _NT_SYMBOL_PATH=C:\DebugSymbols
```

- из командной строки выполняю команду LiveKd -w -Q to start WinDbg;
- когда появится приветствие WinDbg, ввожу следующую команду, чтобы создать полный дамп памяти:

```
.dump /f c:\memory.dmp
```

Предварительно следует убедиться, что на диске достаточно места.

- ввожу q, чтобы выйти из WinDbg, а затем `n`, чтобы выйти из LiveKd.

Теперь полный дамп памяти (C:\memory.dmp) можно сжать и передать для анализа.

Примечание Этот метод создан на основе статьи из блога Карла Харрисона (*http://blogs.technet.com/carlh*).

ListDLLs

ListDLLs — консольная утилита, показывающая информацию о библиотеках DLL, загруженных в процессы на локальном компьютере. Она может показывать DLL, используемые во всей системе или в отдельных процессах, и позволяет искать процессы, загрузившие указанную библиотеку DLL. Ее можно использовать для проверки версии DLL, загруженной процессом, и пути загрузки. Кроме того, она может указывать библиотеки DLL, которые были перемещены из предпочитаемого базового адреса или заменены после загрузки.

ListDLLs требует администраторские права и привилегию Debug (Отладка) только для получения списка DLL из процессов, запущенных от

имени другого пользователя или с более высоким IL. Повышенных разрешений не требуется для работы с процессами, запущенными от имени того же пользователя, и для процессов с таким же или более низким IL.

Синтаксис командной строки ListDLLs таков:

```
listdlls [-r] [имя_процесса | PID | -d имя_dll]
```

При запуске без параметров ListDLLs выводит список всех процессов и библиотек DLL, загруженных в них (рис. 7-18). Для каждого процесса ListDLLs выводит разделитель в виде пунктирной линии, имя и PID процесса. Если ListDLLs имеет разрешения, необходимые для открытия процесса, отображается также полная командная строка, использованная для запуска этого процесса, и библиотеки DLL, загруженные в процесс. ListDLLs сообщает базовый адрес, размер, версию и путь к загруженным DLL в табличном виде с заголовками столбцов. Базовый адрес — адрес виртуальной памяти, по которому загружен модуль. Размер — число смежных байтов, начиная от базового адреса, занимаемых образом DLL. Версия извлекается из ресурса версии файла, если имеется, в противном случае остается пустой. Путь — полный путь к библиотеке.

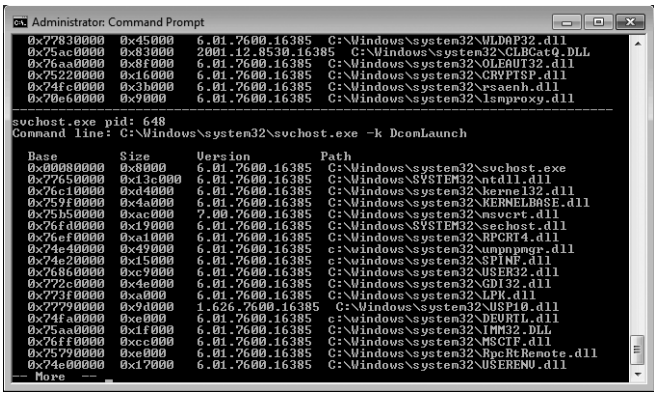


Рис. 7-18. Выходные данные ListDLLs

ListDLLs сравнивает временную отметку из заголовка Portable Executable (PE) образа, расположенного в памяти, с заголовком PE образа на диске. Различие говорит о том, что файл DLL был заменен на диске после того, как процесс загрузил его. ListDLLs сообщает о таких различиях следующим образом:

```
***  Loaded C:\Program Files\Utils\PrivBar.dll  differs from file image:
***  File timestamp: Wed  Feb 10 22:06:51 2010
***  Loaded image timestamp:  Thu Apr 30 01:48:12 2009
*** 0x10000000   0x9c000   1.00.0004.0000  C:\Program Files\Utils\PrivBar.dll
```

ListDLLs сообщает только о библиотеках DLL, загруженных как исполняемые образы. В отличие от функции DLL View утилиты Process Explorer

(см. гл. 3), она не перечисляет библиотеки и другие файлы или проецируемые файлы, загруженные загрузчиком образов как данные, включая DLL, загруженные только для получения ресурсов.

Параметр –r сообщает о библиотеках DLL, которые были перемещены по другому адресу виртуальной памяти из базового адреса, указанного в образе[2]. Если указать этот параметр, перед перемещенной библиотекой DLL в выходных данных будет стоять строка, сообщающая о перемещении и указывающая базовый адрес образа. В следующем примере показана библиотека webcheck.dll, имеющая базовый адрес образа 0x00400000, но загруженная по адресу 0x01a50000:

```
### Relocated from base of 0x00400000:
0x01a50000 0x3d000  8.00.6001.18702 C:\WINDOWS\system32\webcheck.dll
```

Чтобы ограничить процессы, отображаемые в выходных данных, укажите в командной строке имя или PID процесса. Если указать имя процесса, ListDLLs покажет только процессы, имя образа которых совпадает или начинается с указанного текста. Например, чтобы получить список DLL, загруженных всеми экземплярами Internet Explorer, выполните следующую команду:

```
listdlls iexplore.exe
```

ListDLLs покажет все процессы *iexplore.exe* и библиотеки DLL, загруженные в каждый из них. Если указать PID, ListDLLs покажет библиотеки DLL только этого процесса.

Чтобы найти процессы, загрузившие определенную библиотеку DLL, добавьте в командную строку параметр -d, за которым укажите полное или частичное имя библиотеки DLL. ListDLLs выполнит поиск всех процессов, на открытие которых хватает прав, и проверит полный путь каждой из их библиотек DLL. Если указанное имя встречается в любом месте пути загруженной DLL, будет выведена информация об этом процессе и соответствующих DLL. Например, чтобы найти все процессы, загрузившие Crypt32.dll, выполните следующую команду:

```
listdlls -d crypt32
```

Этот параметр можно использовать не только для поиска библиотек DLL по имени, но и для поиска папок. Чтобы получить список всех DLL, загруженных из иерархии папок Program Files, выполните следующую команду:

```
listdlls -d «program files"
```

---

[2] С появлением в Windows Vista рандомизации адресного пространства (Address Space Layout Randomization, ASLR), базовый адрес ASLR-совместимой DLL изменяется после каждой загрузки системы. ListDLLs сообщает о перемещении библиотеки DLL только в том случае, если она загружается в процесс по другому адресу, отличающемуся от предпочитаемого адреса ASLR в текущем сеансе работы, из-за конфликта с другим модулем.

Handle

Handle — консольная утилита, отображающая информацию об описателях объектов, имеющихся у процессов системы. Описатели представляют открытые экземпляры базовых объектов операционной системы, с которыми взаимодействуют приложения, таких как файлы, разделы реестра, примитивы синхронизации и общая память. Утилиту Handle можно использовать для поиска программ, имеющих открытые файлы или папки, предотвращать доступ к ним или их удаление из другой программы. Handle можно также использовать для получения списка имен и типов объектов, открытых определенной программой. Подробнее об описателях объектов см. в разделе «Описатели» главы 2.

Поскольку основное предназначение утилиты Handle — выявление используемых файлов и папок, при ее запуске без параметров командной строки отображается список всех описателей файлов и именованных разделов, занятых процессами. Параметры командной строки позволяют получать список всех типов объектов, искать объекты по имени, показывать информацию только о указанных процессах, отображать счетчики описателей по типам объектов, отображать сведения об объектах-разделах, имеющих поддержку файла подкачки, отображать имя пользователя вместе с информацией об описателях и закрывать открытые описатели (что не рекомендуется).

Заметьте, что при загрузке библиотеки DLL или проецировании файла другого типа в адресное пространство процесса посредством API *LoadLibrary* описатель не добавляется в таблицу описателей процесса. Следовательно, такие файлы могут использоваться, и их нельзя будет удалить, несмотря на то, что поиск описателей может ни к чему не привести. Утилита ListDLLs, о которой говорилось ранее в этой главе, может выявлять библиотеки DLL, загруженные как исполняемые образы. Больше возможностей предоставляет функция поиска утилиты Process Explorer (гл. 3), позволяющая искать библиотеки DLL и имена описателей в одной операции и включающая в поиск библиотеки DLL, загруженные как данные.

Поиск и получение списка описателей

Синтаксис командной строки для получения списка описателей объекта таков:

```
handle [-a [-l]] [-p процесс|PID] [[-u] имя_объекта]
```

Если запустить утилиту без параметров командной строки, она выведет список всех процессов и все описатели файлов и именованных разделов, занятых этими процессами, разделяя данные о каждом процессе пунктирной линией. Для каждого процесса отображается имя, PID и имя учетной записи, от имени которой запущен процесс. Далее указываются описатели, принадлежащие этому процессу. Значения описателей показаны в шестнадцатеричном виде, вместе с типом объекта и именем (если таковое имеется).

Описатели файлов кроме обычных файлов могут включать папки, драйверы устройств и конечные точки связи. Информация об описателе файла включает режим совместного использования, установленный при открытии описателя. Флаги совместного использования заключаются в скобки и могут быть следующими: *R*, *W* и *D*. Они указывают, что другие вызывающие стороны (включая другие потоки того же процесса) могут открывать тот же файл для чтения, записи или удаления, соответственно. Дефис вместо буквы указывает на то, что режим совместного использования не установлен. Если не установлен ни один флаг, объект открывается для эксклюзивного использования через его описатель.

Именованные разделы, также называемые *объектами-проекциями*, могут находиться в файлах на диске или в файле подкачки. Открытый описатель спроецированного файла может помешать удалению этого файла. Именованные разделы из файла подкачки нужны для создания общих областей памяти для разных процессов.

Чтобы найти описатели объекта по имени, добавьте имя объекта в командную строку. Handle покажет список всех описателей объектов, содержащих указанное имя. Поиск выполняется без учета регистра. При поиске объекта по имени можно добавить параметр –u, чтобы отображались также имена учетных записей пользователей, от имени которых работают процессы, владеющие перечисленными описателями.

Поиск объекта по имени изменяет формат выходных данных. Вместо группировки описателей по процессу и использования разделителей, каждая строка будет содержать имя процесса, PID, тип объекта, значение описателя, имя описателя и, если указано, имя пользователя.

Например, если нужно найти процесс, использующий файл MyDataFile.txt из папки MyDataFolder, можно выполнить следующую команду:

```
handle mydatafolder\mydatafile.txt
```

Чтобы просмотреть все типы описателей, а не только описатели файлов и именованных разделов, добавьте параметр –a. Будет показан список всех описателей объектов всех типов, включая объекты без имени. Можно комбинировать параметры –a и –l (строчная L), чтобы получить список всех объектов-разделов и размеров распределений файла подкачки (если таковые имеются), связанных с каждым из них. Это может облегчить поиск утечек памяти, вызванных проецируемыми разделами из файла подкачки.

Чтобы ограничить список процессов, включенных в выходные данные, добавьте параметр –p, за которым укажите частичное или полное имя процесса или его идентификатор. Если указано имя процесса, Handle покажет список описателей процессов, чье имя образа совпадает или начинается с указанного текста. Если указать PID, будет показан список описателей только этого процесса.

Рассмотрим примеры. Следующая команда выводит список описателей файлов и именованных разделов, которыми владеют процессы, чье имя начинается с «explore», включая все запущенные экземпляры Explorer.exe:

```
handle -p explore
```

Часть выходных данных этой команды показана на рис. 7-19.

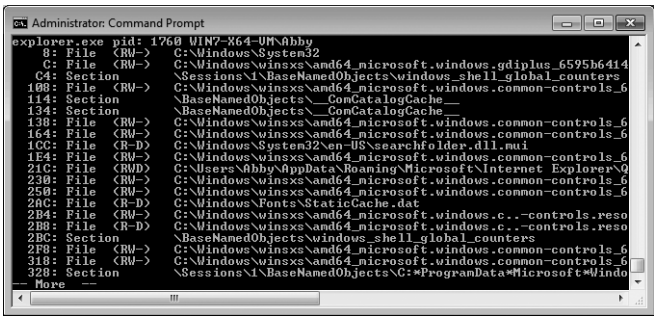


Рис. 7-19. Часть выходных данных команды handle -p explore

В отличие от предыдущей, следующая команда показывает список описателей объектов всех типов и для всех процессов, где имя объекта содержит слово «explore».

```
handle -a explore
```

На рис. 7-20 показана часть выходных данных этого поиска по имени объекта, включающих процессы, имеющие описатели файлов, разделов реестра, процессов и потоков с текстом «explore» в имени.

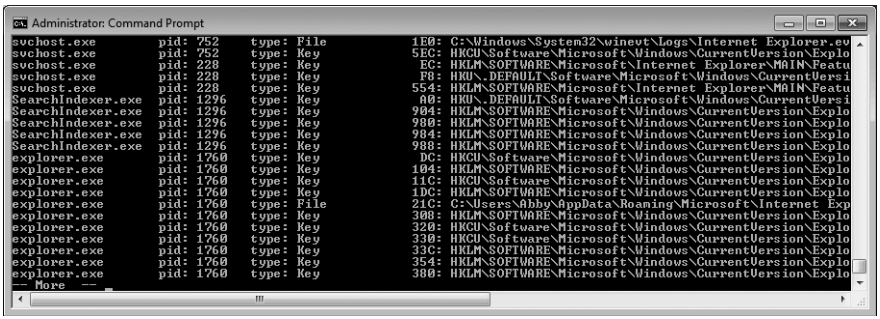


Рис. 7-20. Часть выходных данных команды handle -a explore

В следующем примере продемонстрирован поиск имени объекта, содержащего пробел, с включением имени пользователя в выходные данные. Отображаются объекты всех типов, содержащие искомое имя, включая разделы реестра, имя которых начинается с буквы «c»:

```
handle -a -p c -u «session manager»
```

Выходные данные этой команды показаны на рис. 7-21.

Для запуска Handle требуются администраторские права. Так как некоторые объекты предоставляют полный доступ только системе, но не администраторам, обычно можно получить более полные данные, запустив Handle от имени системы, используя PsExe (см. главу 6). Если Handle.exe и PsExec

находятся в папках, включенных в системную переменную PATH, можно использовать следующую команду:

```
psexec -s handle -accepteula -a
```

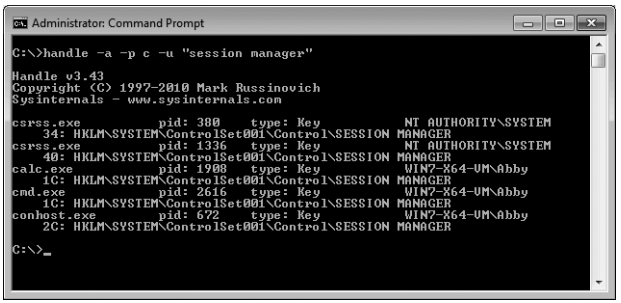


Рис. 7-21. Выходные данные команды handle -a -p c -u «session manager»

Счетчики описателей

Чтобы посмотреть, сколько объектов каждого типа открыто, используйте параметр –s. Будет показан список объектов всех типов, для которых в системе имеются любые открытые описатели, и число описателей каждого объекта. В конце списка будет показано общее число описателей.

Чтобы получить информацию только для определенных процессов, добавьте параметр –p, за которым укажите полное или частичное имя процесса или идентификатор:

```
handle -s [-p процесс|PID]
```

Используя алгоритм сопоставления имен процессов, описанного в разделе «Поиск и получение списка описателей», Handle показывает число описателей объектов, имеющихся у указанного процесса или процессов и у объектов определенного типа, а в конце подсчитывает общее число описателей. Следующая команда отображает список счетчиков описателей для всех процессов Explorer в системе:

```
handle -s -p explorer
```

Выходные данные будут следующими:

```
Handle type summary:
  ALPC Port       : 44
  Desktop         : 5
  Directory       : 5
  EtwRegistration : 371
  Event           : 570
  File            : 213
  IoCompletion    : 4
  Key             : 217
  KeyedEvent      : 4
```

```
  Mutant          : 84
  Section         : 45
  Semaphore       : 173
  Thread          : 84
  Timer           : 7
  TpWorkerFactory : 8
  UserApcReserve  : 1
  WindowStation   : 4
  WmiGuid         : 1
Total handles: 1840
```

Закрытие описателей

Как говорилось ранее, процесс может освободить описатель на объект, когда этот объект больше не нужен ему, а оставшиеся описатели будут закрыты, когда процесс завершится. При помощи Handle можно закрывать описатели, открытые процессом, не завершая этот процесс. Обычно это рискованно. Поскольку процесс, владеющий описателем, не знает о том, что этот описатель закрыт, использование этой функции может привести к повреждению данных или сбою в приложении; закрытие описателя в системном процессе или критически важном процессе пользовательского режима, таком как Csrss, может привести к сбою в системе. Кроме того, при последующем распределении ресурсов тем же процессом может быть получено старое значение описателя, так как оно больше не используется. Если программа попытается обратиться к уже закрытому объекту, она может получить не тот объект, который ожидает.

Помните об этом, когда решите закрыть описатель. Синтаксис командной строки для закрытия описателя следующий:

```
handle -c значение_описателя -p PID [-y]
```

Значение описателя интерпретируется как шестнадцатеричное число, а владеющий процесс должен быть задан по PID. Прежде чем закрыть описатель, Handle показывает информацию о нем, включая его тип и имя, и запрашивает подтверждение. Подтверждение можно пропустить, добавив в командную строку параметр –y.

Windows защищает некоторые описатели объектов, так что их нельзя закрыть, за исключением случаев с принудительным завершением процесса. Попытки закрытия таких описателей заканчиваются неудачей без каких-либо уведомлений, и Handle сообщает, что описатель закрыт, хотя на самом деле это не так.

Глава 8

Утилиты системы безопасности

Предмет данной главы — пакет утилит Sysinternals, предназначенных для настройки и управления системой безопасности ОС Windows:

- **SigCheck** — консольная программа для проверки подлинности цифровых подписей файлов, вывода списков хеш-значений и информации о версиях файлов.
- **AccessChk** — консольная программа для поиска объектов (файлов, разделов реестра и служб), предоставляющих разрешения отдельным пользователям или группам, а также для вывода подробной информации о предоставленных разрешениях.
- **AccessEnum** — GUI-программа, выполняющая поиск файлов и разделов реестра и выявляющая места возможного изменения разрешений.
- **ShareEnum** — GUI-программа, которая перечисляет общие файлы и принтеры, доступные через сеть, а также пользователей, имеющих к ним доступ.
- **ShellRunAs** — это расширение оболочки, восстанавливающее возможность запуска программы под другой учетной записью пользователя в Windows Vista.
- **Autologon** — GUI-программа, которая настраивает учетную запись пользователя для автоматического входа при загрузке системы.
- **LogonSessions** — консольная программа, которая перечисляет активные сеансы LSA на компьютере.
- **SDelete** — консольная программа, которая гарантированно удаляет с жесткого диска файлы и папки, а также стирает данные в нераспределенных областях жесткого диска.

SigCheck

SigCheck — это многофункциональная консольная программа, которая решает различные задачи по защите файлов и папок. Главная ее задача состоит в проверке наличия у файлов доверяемого сертификата цифровой подписи. Как показано на рис. 8-1, SigCheck также способна выводить информацию о каталогах и владельцах подписей, вычислять хеш для файлов по разным

алгоритмам, а также отображать расширенную информацию о версиях файлов. Кроме того, программа может отображать внедренный манифест файла, искать в папках неподписанные файлы и выводить результаты в формате CSV.

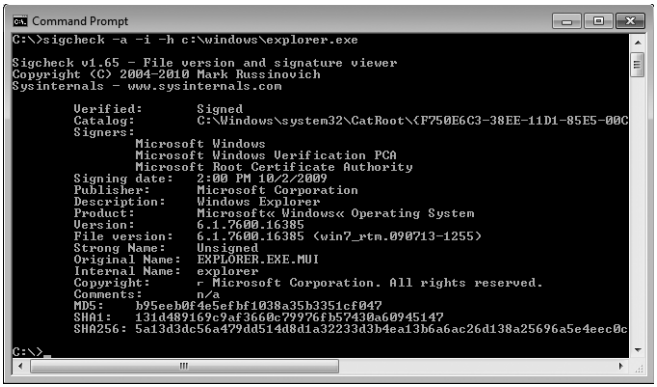


Рис. 8-1. Вывод команды sigcheck –a –I –h c:\windows\explorer.exe

Цифровая подпись файла помогает проверить его подлинность и целостность. Проверенная подпись свидетельствует о том, что файл не был изменен с момента подписания. Надежность этого свидетельства, обеспеченная сертификатом цифровой подписи, во многом зависит от разборчивости центра сертификации (ЦС) при выдаче сертификатов, удостоверяющего подлинность владельца, бдительности владельца сертификата при хранении сертификата, а также от системы проверки подлинности, препятствующей установке неавторизованных сертификатов корневых ЦС.

Одна из статей расходов при ведении бизнеса включает предоставление клиентам гарантий подлинности программной продукции. Для этого многие официальные издатели ПО приобретают сертификаты для цифровой подписи у официального ЦС, такого как VeriSign или Thawte. Такие сертификаты позволяют подписывать файлы, устанавливаемые на компьютеры клиентов. Отсутствие действительной подписи у исполняемого файла, пришедшего якобы от известной фирмы, должно вызывать подозрение.

Примечание В прошлом вредоносные программы подписывали редко. С тех пор их издатели стали действовать более изощренно. Некоторые авторы вредоносного ПО даже создают подставные организации и приобретают через них сертификаты для цифровой подписи у официальных ЦС. Другие же просто крадут плохо защищенные закрытые ключи у компаний — законных владельцев сертификатов, чтобы подписывать ими вредоносное ПО.

Параметры командной строки SigCheck предоставляют множество возможностей при проверке подлинности, в том числе выбор файлов для проверки и форматированный вывод данных (см. синтаксис ниже и табл. 8-1).

```
sigcheck.exe [-e] [-s] [-i] [-r] [-u] [-c ЦСtalogFile] [-a] [-h] [-m] [-n]
[-v] [-q] target
```

Табл. 8-1. Параметры командной строки SigCheck

Параметр	Описание
Target	Указывает процессу файл или каталог. Может содержать подстановочные знаки
Проверка подписи	
-i	Отображает имя каталога и владельцев подписей образов
-r	Проверяет сертификат на отзыв
-u	Отображает только неподписанные файлы, включая файлы с недействительными подписями
-c	Выполняет поиск подписи в указанном файле
Проверка файлов	
-e	Проверяет только исполняемые файлы (выявляются по заголовкам, а не по расширениям)
-s	Выполняет рекурсивный обход вложенных папок
Дополнительная информация о файлах	
-a	Отображает дополнительную информацию о версиях файлов
-h	Отображает хеш файлов
-m	Отображает манифест
-n	Отображает только номер версии файла
Формат вывода	
-v	Вывод данных в формате CSV (не совместим с -i и -m)
-q	«Тихий» режим (не выводит заголовок)

Параметр *target* — единственный обязательный параметр. Он может указывать отдельный файл (например, explorer.exe), несколько файлов с помощью подстановочных знаков (например, *.dll) или папку, используя для этого относительные или абсолютные пути. Если указать папку, то SigCheck проверит все файлы в этой папке.

Проверка подлинности подписи

При отсутствии дополнительных параметров SigCheck отображает следующую информацию о каждом из проверяемых файлов:

- **Verified (Проверен)** Если файл подписан сертификатом цифровой подписи, унаследованным от корневого центра сертификации, доверяемого на этом компьютере, и не был изменен с момента подписания, то в этом поле отображается надпись **Signed (Подписан)**. Если же файл не подписан, в поле отображается надпись **Unsigned (Не подписан)**. Если файл

подписан, но подпись не прошла проверку, появляется сообщение об этом. Возможные причины: сертификат недействителен на момент подписания; корневой ЦС считается ненадежным (такое возможно, например, в случае самоподписанного сертификата); файл был изменен после подписания.

- **Signing date (Дата подписания)** Отображает дату подписания файла. Если файл не подписан, отображается строка *n/a*.
- **Publisher (Издатель)** Название компании из внедренных сведений о версии файла (если есть).
- **Description (Описание)** с информацией из внедренных сведений о версии файла (если есть).
- **Product (Продукт)** Поле **Product Name** из внедренных сведений о версии файла (если есть).
- **Version (Версия)** Поле **Product Version** из внедренных сведений о версии файла (если есть). Заметьте, что эти данные берутся из строковых, а не двоичных данных, используемых для сравнения версий.
- **File version (Версия файла)** Поле **File Version** из внедренных сведений о версии файла (если есть). Обратите внимание на то, что эти данные также взяты из строкового раздела ресурса версии, а не из двоичного значения, используемого для сравнения версий.

Для отображения дополнительных данных подписи добавьте в командную строку параметр -i. Если подпись файла действительна, то с помощью данного параметра можно увидеть два дополнительных поля:

- **Catalog (Каталог)** Показывает файл, в котором хранится подпись. В большинстве случаев указанным файлом будет подписанный файл, однако если подпись содержится не в самом файле, в каталоге, отображается файл каталога (CAT-файл). Многие файлы, которые поставляются с Windows, подписаны в CAT-файле. Подписание в CAT-файле может в некоторых случаях повысить производительность, но лучше использовать этот способ для подписания неисполняемых файлов в форматах, не поддерживающих внедрение подписи.
- **Signers (Владельцы подписей)** Отображает общее имя (CN) в строке **Subject (Субъект)** сертификата цифровой подписи и сертификатах ЦС в его цепочке.

По умолчанию SigCheck не проверяет, был ли сертификат подписи отозван издателем. Чтобы проверить это, добавьте в командную строку параметр -r . Имейте в виду, что проверка отзыва может потребовать намного больше времени для проверки подписи, так как SigCheck приходится запрашивать точки распространения списка отзывов сертификатов (CRL).

Чтобы выполнить поиск только неподписанных файлов, добавьте в командную строку параметр -u. После этого SigCheck проверит все указанные файлы, но покажет информацию только о неподписанных файлах либо файлах, подписи которых невозможно проверить.

Windows ведет базу данных каталогов подписей для быстрого поиска информации о подписи по хеш-значению файла. Если вы хотите проверить файл каталога, не зарегистрированный в базе данных, укажите его в командной строке с параметром -c.

Проверяемые файлы

Большинство неисполняемых файлов не имеет цифровой подписи. Тем не менее, некоторые неизменяемые файлы неисполняемых форматов, поставляемые с Windows, могут быть подписаны в CAT-файле. Однако изменяемые файлы данных, в том числе, файлы инициализации, резервные файлы кустов реестра, документы и временные файлы, никогда не имеют зашифрованной подписи. При проверке папки с множеством подобных файлов трудно найти среди них неподписанные исполняемые файлы, которые обычно и являются целью поиска. Чтобы отсеять лишнее, сначала ищут файлы по маске **.exe,* затем **.dll, *.ocx, *.scr* и т.д. Главный недостаток данного метода не в лишней работе по поиску и не в риске пропустить какое-либо расширение, а в том, что исполняемые файлы, которым преднамеренно назначено расширение .tmp или любое другое расширение, нетипичное для исполняемого файла, либо не имеющие никакого расширения, будут пропущены при поиске. Создатели вредоносного ПО часто прячут его за безопасными с виду расширениями файлов.

Поэтому вместо перебора расширений добавьте в командную строку SigCheck параметр -e, который заставит программу проверить только исполняемые файлы. В частности, SigCheck проверяет, начинается ли файл со строки *MZ*. Заголовки всех 16-, 32- и 64-разрядных исполняемых файлов Windows, включая приложения, DLL и системные драйверы, начинаются с этих букв. SigCheck игнорирует расширения файлов, так что исполняемые файлы, замаскированные другими расширениями, не ускользнут от проверки.

Для поиска в иерархии папок добавьте в командную строку параметр -s. SigCheck будет проверять файлы, соответствующие параметру *target,* в папке, указанной данным параметром (либо в текущей папке, если параметр *target* не указывает папку) и во всех вложенных в нее папках. Приведенная ниже команда проверяет все файлы с расширением *.dll в папке C:\Program Files и ее вложенных папках:

```
sigcheck -s «c:\program files\*.dll»
```

Дополнительная информация о файле

Добавьте в командную строку параметр -a, чтобы извлечь дополнительную информацию из каждого проверенного файла. При добавлении -a вывод SigCheck дополняется следующими полями:

- **Strong Name (Строгое имя)** Если файл является сборкой .NET и имеет подпись и строгое имя, в этом поле отображается надпись **Signed**, в противном случае — **Unsigned** (подписание сборок .NET с назначением им

строгого имени не связано с цифровой подписью на основе сертификата и не соответствует какому-либо уровню доверия).

- **Original Name (Исходное имя)** Поле **Original Name** из внедренных сведений о версии файла (если есть).
- **Internal Name (Внутреннее имя)** Поле Internal Name из внедренных сведений о версии файла (если есть).
- **Copyright (Авторское право)** Поле **Copyright** из внедренных сведений о версии файла (если есть).
- **Comments (Комментарии)** Поле **Comments** из внедренных сведений о версии файла (если есть).

Хеш — это статистически уникальное значение, сгенерированное по блоку данных с использованием такого криптографического алгоритма, который при малейшем изменении данных приводит к генерации совершенно иного хеша. При использовании надежного хеш-алгоритма на современном уровне технологий практически невозможно изменить данные, оставив неизменным хеш, поэтому хеш-значения применяют для выявлении случайных повреждений или несанкционированной модификации данных. При добавлении в командную строку параметра -h SigCheck вычисляет и отображает хеш для проверяемых файлов по алгоритму MD5, SHA1 или SHA256. Эти хеши можно сравнивать с хеш-значениями, рассчитанными для заведомо целых данных, чтобы убедиться в целостности проверяемого файла. С помощью хеш-значения можно проверить и неподписанные файлы при наличии заведомо целой версии последних. Некоторые системы проверки файлов проверяют именно хеш-значения, а не подписи.

Манифест приложения — это документ в формате XML, который можно внедрять в файл программы. Впервые их использовали в Windows XP, где они помогали различать различные версии одних и тех е сборок. В Windows Vista и Windows 7 схема манифеста была дополнена возможностями проверки совместимости с разными версиями Windows, а также требования прав администратора. Наличие манифеста, совместимого с Windows Vista, также отключает виртуализацию файла и реестра для процесса. Для получения дампа внедренного в файл манифеста добавьте в командную строку параметр -m. Ниже приводится вывод SigCheck с собственным манифестом этой программы:

```
c:\program files\sysinternals\sigcheck.exe:
        Verified:   Signed
        Signing date: 19:14 6/7/2010
        Publisher:  Sysinternals - www.sysinternals.com
        Description: File version and signature viewer
        Product: Sysinternals Sigcheck
        Version: 1.70
        File version: 1.70
        Manifest:
```

```
<assembly xmlns=»urn:schemas-microsoft-com:asm.v1» manifestVersion=»1.0»>
   <trustInfo xmlns=»urn:schemas-microsoft-com:asm.v3»> <security>
      <requestedPrivileges>
         <requestedExecutionLevel level=»asInvoker» uiAccess=»false»></
requestedExecutionLevel>
      </requestedPrivileges>
      </security>
   </trustInfo>
</assembly>
```

Для того чтобы вывести только номер версии файла, добавьте в командную строку параметр -n. SigCheck отображает только значение поля **File Version** из внедренного в файл ресурса с версией файла (если таковой имеется), в противном случае выводится строка *n/a*. Этот параметр полезен для использования в командных файлах, лучше использовать его при проверке конкретных файлов.

Разумеется, параметры командной строки можно сочетать. Например, приведенная ниже команда выполняет поиск в папке system32 и вложенных в нее папках неподписанных исполняемых файлов с выводом хеш-значения и подробной информации о версии:

```
sigcheck -u -s -e -a -h c:\windows\system32
```

Формат вывода данных

Как правило, SigCheck выводит данные в виде форматированного списка (рис. 8-1). Чтобы получить данные в формате CSV для импорта в электронную таблицу или базу данных добавьте в командную строку параметр –v. SigCheck выводит заголовки столбцов в соответствии с информацией о файле, запрошенной посредством других параметров командной строки. За именами столбцов следует строка значений, разделенных запятой, для каждого проверяемого файла. Имейте в виду, что параметр -v не может использоваться совместно с параметрами -i и -m.

Вывод заголовка SigCheck можно отключить с помощью параметра -q. Удаление этих строк облегчает обработку вывода SigCheck в командных файлах и вывод CSV.

AccessChk

AccessChk — это консольная программа, которая отображает действующие разрешения на доступ к защищенным объектам, права учетных записей пользователей и групп, а также подробности маркера процесса. Программа может просматривать вложенные папки и разделы реестра, содержащие объекты, для которых у пользователей или групп есть (или, наоборот, отсутствуют) разрешения на доступ для чтения или записи. Также программа может отображать неструктурированный список управления доступом к защищенным объектам.

Что такое «действующие разрешения»?

Действующие разрешения — это разрешения на доступ к объекту, которыми обладает пользователь или группа в зависимости от принадлежности к той или иной группе, а также разрешения, в которых может быть отказано явно. В качестве примера рассмотрим папку C:\Documents and Settings на компьютере под управлением Windows 7, где эта папка представлена символической ссылкой, обеспечивающей совместимость приложений. Разрешением «полный доступ» для нее обладают группа **Администраторы** и процесс System, а также группа Everyone (Все) — разрешения на доступ для чтения. При этом у группы **Все** явно отозвано разрешение List Folder (Содержимое папки) для этой папки. Соответственно, если пользователь MYDOMAIN\Abby является членом группы **Администраторы**, его действующие разрешения включают все разрешения, кроме **Содержимое папки**. Если же MYDOMAIN\Abby является обычным пользователем и, следовательно, членом группы **Все**, его действующие разрешения содержат только разрешение на чтение без возможности просмотра содержимого папки.

В Windows имеется средство Effective Permissions (Действующие разрешения), доступное в окне Advanced Security Settings, которое открывается щелчком кнопки Advanced в редакторе разрешений для некоторых типов объектов. Средство Effective Permissions проверяет и отображает действующие разрешения на доступ к выбранному объекту для указанного пользователя или группы. AccessChk использует те же API-функции, что и Windows, и может выполнять те же операции, но для гораздо более широкого разнообразия объектов, причем ее можно использовать в сценариях. AccessChk может отображать разрешения на доступ к файлам, папкам, разделам реестра, процессам, а также к любым типам объектов, объявленным в пространстве имен Диспетчера объектов, включая каталоги, разделы реестра и семафоры.

Имейте в виду, что действующие разрешения в Windows — это всего лишь приблизительный набор реальных разрешений, которыми может обладать пользователь, выполнивший вход в систему. Реальные разрешения могут отличаться, поскольку они предоставляются или отзываются на основании способа входа пользователя (под записью интерактивного пользователя или службы). При определении же действующих разрешений способ входа в систему не учитывается. При определении разрешений на доступ к удаленным объектам разрешения на общий доступ, членство в локальных группах и их привилегии не учитываются. Кроме того, есть аномалии, связанные с включением или исключением встроенных локальных групп (см. статью Knowledge Base по адресу *http://support.microsoft.com/kb/323309)*. В частности, я недавно обнаружил новую ошибку в определении разрешений для группы Администраторы. Наконец, действующие разрешения могут зависеть от наличия прав у записи, под которой определяются разрешения, доступа к информации о проверяемом объекте в Active Directory (см. статью Knowledge Base под номером 331951 по адресу *http://support.microsoft.com/kb/331951)*.

Использование AccessChk

Базовый синтаксис AccessChk имеет следующий вид:

```
accesschk [параметры] [имя_пользователя_или_группы] имя_объекта
```

Параметр *имя_объекта* — это защищенный объект, который нужно проверить. Если объект является контейнером как, например, папка файловой системы, или раздел реестра, AccessChk покажет все объекты в данном контейнере, а не сам контейнер. Если указать параметр *имя_пользователя_или_группы*, AccessChk покажет действующие разрешения для данного пользователя или группы, в противном случае программа будет отображать набор действующих прав для всех учетных записей, содержащихся в списке управления доступом (ACL) объекта.

По умолчанию параметр *имя_объекта* интерпретируется как объект файловой системы и может включать подстановочные знаки ? и *. Если объект является папкой, AccessChk показывает действующие разрешения на доступ ко всем вложенным в нее файлам и папкам. Если объект является файлом, AccessChk отображает действующие разрешения на доступ к нему. Ниже в качестве примера приводятся действующие разрешения на доступ к c:\windows\explorer.exe на компьютере под управлением Windows 7:

```
c:\wi ndows\explorer.exe
   RW NT SERVICE\TrustedInstaller R
   BUILTIN\Administrators R NT
   AUTHORITY\SYSTEM R
   BUILTIN\Users
```

AccessChk суммирует разрешения на доступ ко всем отображаемым объектам для каждого пользователя или группы из ACL. Если учетная запись обладает разрешением на чтение, выводится буква R, а если разрешением на запись — W; если разрешения отсутствуют, не выводится ничего.

Именованные каналы считаются объектами файловой системы. Используйте префикс “\pipe\”, чтобы указать путь к именованному каналу, или просто “\pipe\”, чтобы указать контейнер, в котором находятся все именованные каналы: команда **accesschk \pipe** показывает действующие разрешения на доступ ко всем именованным каналам на компьютере, а **accesschk \pipe\srvsvc** — действующие разрешения на доступ к каналу srvsvc, если он существует. Имейте в виду, что поиск с подстановками, например **\pipe\s***, не поддерживается из-за ограничений Windows.

Тома также считаются объектами файловой системы. Для проверки локальных томов служит синтаксис \\.\X:, где X — буква диска. Например, команда **accesschk \\.\C:** показывает разрешения на доступ к тому C. Заметьте, что разрешения на доступ к тому — это не то же самое, что разрешения на доступ к его корневому каталогу. Разрешения на доступ к тому определяют, кто может выполнять задачи по обслуживанию тома с использованием, например, дисковых утилит, описанных в Главе 12.

Параметр *параметры* позволяет задавать типы объектов и разрешений, необходимость рекурсивного обхода иерархии контейнеров, выводимые данные, а также необходимость отображения действующих разрешений или ACL объекта. Список параметров приводится в табл. 8-2, подробнее о них — ниже.

Табл. 8-2. Параметры командной строки AccessChk

Параметр	Описание
Указанный тип объекта	
-d	Задан контейнер и вывод разрешений на доступ к нему без права доступа к его содержимому
-k	Раздел реестра
-c	Служба Windows
-p	PID, полное или частичное имя процесса
-f	В сочетании с -p отображает полную информацию маркера указанного процесса
-o	Объект в пространстве имен Диспетчера объектов Windows
-t	В сочетании с -o указывает тип объекта, в сочетании с -p — уведомляет о разрешениях на доступ к потокам процесса
-a	Право учетной записи
Поиск прав доступа	
-s	Выполняет рекурсивный обход иерархии контейнера
-n	Отображает только объекты, не предоставляющие прав доступа (обычно в сочетании с *имя_пользователя_или_группы*)
-w	Отображает только объекты, предоставляющие право на запись
-r	Отображает только объекты, предоставляющие право на чтение
-e	Отображает только объекты с явно заданными IL (только в Windows Vista и выше)
Вывод данных	
-1	Отображает ACL вместо действующих разрешений
-u	Не показывает ошибки
-v	Вывод подробных данных
-q	«Тихий» режим (не показывает заголовок)

Тип объекта

Как сказано выше, если именованный объект является контейнером, например, папкой файловой системы, разделом реестра или каталогом Диспетчера объектов, AccessChk отображает объекты из данного контейнера, а не сам контейнер. Чтобы программа AccessChk показывала разрешения для контейнера, добавьте в командную строку параметр -d. Например, **accesschk c:**

windows показывает действующие разрешения на доступ к каждому файлу и вложенной папке в папке Windows, а **accesschk -d c:\windows** — разрешения на доступ к самой папке Windows. Аналогично, **accesschk** показывает разрешения на доступ к содержимому текущей папки, а **accesschk -d** — разрешения на доступ к самой текущей папке. Наконец, **accesschk** * показывает разрешения на доступ ко всем объектам в текущей папке, тогда как **accesschk -d** * — только разрешения на доступ к объектам из папки, вложенной в текущую папку.

Для проверки разрешений на доступ к разделу реестра добавьте в командную строку параметр -k. Можно указать краткое или полное имя корневого раздела (например, HKLM или HKEY_LOCAL_MACHINE), а также поставить двоеточие (:) после корневого раздела, как делается в Windows PowerShell (подстановочные знаки не поддерживаются). Ниже показаны равноценные команды для вывода разрешений на доступ к подразделам HKLM\Software\Microsoft:

```
accesschk -k hklm\software\microsoft

accesschk -k hklm:\software\microsoft

accesschk -k hkey_local_machine\software\microsoft
```

Для отображения разрешений доступа только к HKLM\Software\Microsoft без доступа к подразделам добавьте параметр -d .

Для отображения разрешений доступа только к службам Windows добавьте в командную строку параметр -c . Укажите * вместо имени объекта, чтобы вывести сведения для всех служб, или **scmanager,** чтобы проверить разрешения Диспетчера управления службами (неполные имена и подстановочные знаки не поддерживаются). Например, **accesschk -c lanmanserver** показывает разрешения на доступ к службе Server на компьютере под управлением Windows 7. Вот вывод этой команды:

```
lanmanserver
   RW NT AUTHORITY\SYSTEM RW
   BUILTIN\Administrators R NT
   AUTHORITY\INTERACTIVE R NT
   AUTHORITY\SERVICE
```

Эта команда показывает разрешения, явно предоставленные для каждой службы группе Authenticated Users (Прошедшие проверку):

```
accesschk -c «authenticated users» *
```

Для служб W может обозначать разрешения на запуск, остановку, перезапуск, изменение конфигурации, а R —разрешения на запрос конфигурации и состояния.

Для просмотра разрешений на доступ к процессам добавьте в командную строку параметр -p. Именем объекта в этом случае может быть ID процесса

(PID) либо имя процесса, например, “explorer”. AccessChk может найти процесс по части его имени: команда **accesschk -p exp** будет показывать разрешения на доступ к процессам с именами, начинающимися на «exp”, включая все экземпляры Explorer. Если указать * в качестве имени объекта, вы получите разрешения на доступ ко всем объектам. Имейте в виду, что для просмотра разрешений на доступ к процессам, работающим под другой учетной записью или с повышенными привилегиями, требуется наличие прав администратора. Вывод команды **accesschk -p 3048** для экземпляра Cmd.exe с повышенными привилегиями на компьютере под управлением Windows 7 выглядит следующим образом:

```
[3048] cmd.exe
   RW BUILTIN\Administrators
   RW NT AUTHORITY\SYSTEM
```

Используйте -p в сочетании с -t для просмотра разрешений на доступ ко всем потокам указанного процесса (имейте в виду, что в командной строке параметр t должен следовать за p). Для того же экземпляра Cmd.exe с повышенными привилегиями команда **accesschk -pt 3048** выводит следующие данные:

```
[3048] cmd.exe
   RW BUILTIN\Administrators
   RW NT AUTHORITY\SYSTEM
   [3048:7148] Thread
   RW BUILTIN\Administrators
   RW NT AUTHORITY\SYSTEM
   R Win7-x86-VM\S-1-5-5-0-248063-Abby
```

У процесса имеется один поток с ID 7148 и разрешениями, аналогичных таковым его процесса-контейнера.

Для просмотра всех сведений маркера процесса используйте -p в сочетании с -f. Для каждого из перечисленных процессов AccessChk покажет разрешения на доступ к маркеру процесса, а затем отобразит пользователя, группы, флаги групп и привилегии маркера.

Разрешения на доступ к объектам из пространства имен Диспетчера объектов (таким как события, семафоры, разделы и каталоги) просматривают с помощью параметра -o. Для вывода данных только об объектах конкретного типа добавьте в команду параметр -t и укажите тип объекта. Например, приведенная ниже команда показывает действующие разрешения на доступ ко всем объектам в каталоге \BaseNamedObjects:

```
accesschk -o \BaseNamedObjects
```

А действующие разрешения только для объектов Section в каталоге \BaseNamedObjects выводит следующая команда:

```
accesschk -o -t section \BaseNamedObjects
```

Если имя объекта не указано, ведется поиск в корне пространства имен. Утилита WinObj, описанная в главе 14, отображает графическое представление пространства имен Диспетчера объектов.

Хотя списки привилегий и прав учетной записи сами по себе не являются защищаемыми объектами, AccessChk может показывать их с помощью параметра -a. Привилегии, в отличие от прав, предоставляют учетной записи возможности выполнения действий в системе в целом, а не над конкретным объектом. В качестве примера можно привести SeBackupPrivilege, которая позволяет учетной записи читать объект в обход контроля доступа. Права же учетной записи определяют, кто и как может выполнять вход в систему, а кто не может. Например, право SeRemoteInteractiveLogonRight необходимо для входа в систему через удаленный рабочий стол. В отличие от прав учетных записей, перечень привилегий отображается в маркерах доступа.

Ниже я расскажу об использовании параметра -a на примерах. Имейте в виду, что для использования этого параметра AccessChk требует наличия прав администратора. Используйте * в качестве имени объекта, чтобы получить полный список привилегий, прав и учетных записей, которым они предоставлены:

```
accesschk -a *
```

Имя учетной записи, за которым следует *, перечисляет все привилегии и права, присвоенные данной учетной записи. Например, следующая команда отображает права и привилегии, присвоенные группе Опытные пользователи (интересно сравнить эти результаты на компьютерах под управлением Windows XP и Windows 7):

```
accesschk -a «power users» *
```

Наконец, указав имя привилегии или права, вы получите список обладающих им учетных записей (узнать имена привилегий и прав можно с помощью команды **accesschk -a** *). Эта команда перечисляет все учетные записи, которым предоставлена привилегия **SeDebugPrivilege:**

```
accesschk -a sedebugprivilege
```

Поиск прав доступа

Одной из наиболее эффективных функций AccessChk является поддержка поиска объектов, предоставляющих право доступа конкретным пользователям или группам. Например, AccessChk позволяет проверить, разрешено ли пользователям изменять какой-либо объект в папке Program Files или найти пользователей с доступом для записи к некоторым службам.

Параметр -s заставляет AccessChk выполнять рекурсивный поиск в контейнерах (папках, разделах реестра или каталогах пространства имен объектов). Параметр -n перечисляет объекты, не предоставляющие прав доступа указанной учетной записи. Параметр -r перечисляет объекты, предоставля-

ющие разрешение на чтение, а параметр -w — объекты, предоставляющие разрешение на запись. Наконец, в Windows Vista и выше параметр -e отображает объекты с явно заданным, а не установленным по умолчанию IL (последнему соответствуют метки Medium и No-Write-Up).

Рассмотрим примеры:

- Поиск в папке Windows объектов, которые разрешено изменять членам группы **Пользователи**:

```
accesschk -ws Users %windir%
```

- Поиск глобальных объектов, доступных для изменения членам группы **Все**:

```
accesschk -wo everyone \basenamedobjects
```

- Поиск разделов реестра внутри HKEY_CURRENT_USER с явно заданным IL:

```
accesschk -kse hkcu
```

- Поиск служб, для которых у группы **Прошедшие проверку** есть разрешения на запись:

```
accesschk -cw «Authenticated Users» *
```

- Перечисление именованных каналов, доступных всем пользователям разрешения для записи:

```
accesschk -w \pipe\*
```

- Перечисление всех объектов из каталога \sessions Диспетчера объектов, не предоставляющих разрешений группе **Администраторы**:

```
accesschk -nos Administrators \sessions
```

В последнем примере мы видим еще одну важную особенность AccessChk. Разумеется, для просмотра разрешений у объекта нужно соответствующее разрешение на чтение. Ясно также, что в системе есть множество объектов, которые запрещено читать обычным пользователям. Например, содержимое профиля пользователя скрыто от других пользователей, не обладающих правами администратора. Чтобы показывать эти объекты, программа AccessChk должна работать с повышенными правами или правами администратора, но и в этом случае ряд объектов доступен не администраторам, а только системе. В случаях, когда маркера администратора недостаточно, AccessChk копирует системный маркер из процесса Smss.exe и, олицетворяя его, повторяет попытку доступа — без этого функции предыдущая команда не действует.

Параметры вывода данных

Видеть можно не только разрешения R и W, но и все разрешения, добавив в командную строку параметр -v. Тогда под именем каждой учетной записью AccessChk перечисляет разрешения, используя символьные име-

на из Windows SDK. Ниже приведены действующие разрешения для %SystemDrive%\, отображаемые с параметром -v на компьютере под управлением Windows 7:

```
C:\
   Medium Mandatory Level (Default) [No-Write-Up]
   RW BUILTIN\Administrators
         FILE_ALL_ACCESS
   RW NT AUTHORITY\SYSTEM
         FILE_ALL_ACCESS R BUILTIN\Users
         FIL E_LIST_DIRECTORY
         FILE_READ_ATTRIBUTES
         FILE_READ_EA
         FILE_TRAVERSE
         SYNCHRONIZE
         READ_CONTROL W NT AUTHORITY\Authenticated Users
   FILE_ADD_SUBDIRECTORY
```

Подробные данные показывают, что администраторы и система имеют полный доступ, пользователи — разрешение на чтение, а аутентифицированные пользователи имеют дополнительную возможность создавать вложенные папки внутри данной папки.

Вместо действующих разрешений можно отображать действующий ACL объекта, используя параметр -l (это строчная буква L). Ниже приведен ACL ссылки “C:\Documents and Settings” в Windows 7 (см. начало раздела об AccessChk). Элементы списка управления доступом (ACE) перечисляется в следующем порядке: записи пользователей и группы, разрешение и запреты, а затем предоставленные и запрещенные разрешения. Если у ACE есть флаги, они отображаются в квадратных скобках с указанием наследования. Если флаг [INHERITED_ACE] отсутствует, данный ACE задан явно.

```
C:\Documents and Settings
   Medium Mandatory Level (Default) [No-Write-Up]
   [0] Everyone
      ACCESS_DENIED_ACE_TYPE
         FIL E_LIST_DIRECTORY
   [1] Everyone
      ACCESS_ALLOWED_ACE_TYPE
         FIL E_LIST_DIRECTORY
         FILE_READ_ATTRIBUTES
         FILE_READ_EA FILE_TRAVERSE
         SYNCHRONIZE READ_CONTROL
   [2] NT AUTHORITY\SYSTEM
      ACCESS_ALLOWED_ACE_TYPE
         FILE_ALL_ACCESS
   [3] BUILTIN\Administrators
      ACCESS_ALLOWED_ACE_TYPE
         FILE_ALL_ACCESS
```

AccessChk сообщает о всех ошибках при перечислении объектов и получении сведений системы безопасности. Добавьте в командную строку параметр -u, чтобы отключить вывод этих сообщений. Скрыть текст «шапки» списка AccessChk позволяет параметр -q.

AccessEnum

AccessEnum — это GUI-программа, облегчающая поиск файлов, папок и разделов реестра с предположительно модифицированными разрешениями. AccessEnum выводит не все разрешения и объекты, а только те, у которых разрешения отличаются от разрешений родительских контейнеров. Так удается выявить точки, в которых набор разрешений был изменен, игнорируя объекты, унаследовавшие данные изменения.

Например, иногда при попытке запустить приложение при отсутствии прав администратора некоторые пользователи предоставляют группе **Все** полный доступ к папке в Program Files, без прав администратора доступной только для чтения. AccessEnum идентифицирует эту папку и показывает, какие пользователи или группы имеют права доступа, отличные от прав доступа к **Program Files** (рис. 8-2). В этом примере первая строка показывает разрешения на доступ к C:\Program Files; вторая строка — вложенную папку, для которой у группы **Все** есть как минимум некоторые разрешения на чтения и запись (а, возможно, и полный доступ); последние две папки не доступны администраторам для записи.

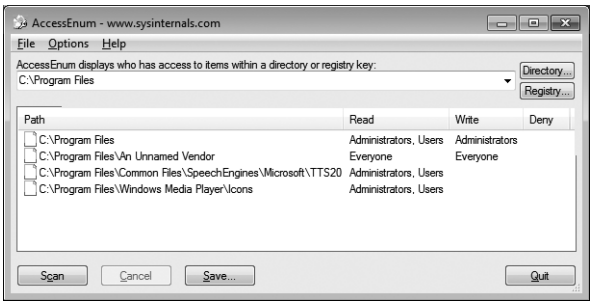


Рис. 8-2. AccessEnum

Введите в текстовое поле вверху окна AccessEnum путь к папке или разделу реестра. Вместо этого можно найти папку, щелкнув кнопку **Directory**, или раздел реестра, щелкнув кнопку **Registry**. Чтобы начать проверку, щелкните кнопку **Scan (Сканировать)**.

AccessEnum анализирует сведения об управлении доступом Windows в поисках записей разрешений на чтение и запись, а также записей отзыва этих разрешений. Найденный объект считается предоставляющим разрешение на запись независимо от того, предоставляет ли он единственное разрешение (например, Write Owner) или полный набор разрешений на запись в рамках полного доступа. Разрешения на чтение обрабатываются аналогичным образом. Имена в столбце **Deny** отображаются, если пользователю или

группе явно отказано в доступе к объекту. Имейте в виду, что группе **Все** запрещен просмотр содержимого «папок»-символических ссылок, описанных в разделе об AccessChk. Программа AccessEnum выводит строку Access Denied, если она не может прочитать дескриптор защиты объекта.

Сравнивая разрешения объекта и его контейнера, утилита AccessEnum проверяет идентичность записей, для которых предоставлены либо отозваны разрешения на чтение или запись. Если для файла предоставлено только разрешение Write Owner, а для его контейнера —только Delete, указанные разрешения считаются равноценными, поскольку допускают внесение изменений (т.е. запись) в объект.

AccessEnum сокращает число отображаемых учетных записей, имеющих доступ к объекту, скрывая записи с разрешениями, дублирующими разрешения, уже назначенные группе, к которой принадлежит учетная запись. Например, если файл доступен для чтения пользователю по имени Боб и группе Маркетинг, и Боб является членом группы Маркетинг**,** то в списке учетных записей, которым разрешено чтение, будет отображаться только группа Маркетинг. Имейте в виду, что в режиме UAC Admin-Approval (только в Windows Vista и выше) процессы, не имеющие повышенных привилегий и запущенные членом группы **Администраторы**, но получившие расширенные права доступа, могут быть не видны. Например, если пользователь Эбби входит в группу Администраторы, AccessEnum покажет объекты, к которым полный доступ есть и у Эбби и у этой группы, как объекты, доступные без ограничений группе **Администраторы**, даже если процессы, запущенные Эбби без повышенных привилегий, также обладают полным доступом к объекту.

По умолчанию AccessEnum отображает только объекты, более доступные по сравнению с их контейнерами. Чтобы увидеть объекты, разрешения на доступ к которым отличаются от таковых к их контейнерам, щелкните меню **Options | File Display Options | Display Files With Permissions That Differ From Parent (Параметры | Параметры отображения файлов | Отображать файлы с разрешениями, отличными от родительских)**.

Поскольку права доступа, предоставленные учетной записи System и другим служебным учетным записям, обычно не представляют интереса, AccessEnum по умолчанию игнорирует их. Чтобы увидеть и эти разрешения, выберите **Show Local System And Service Accounts (Показать учетные записи локальной системы и служб)** в меню **Options**.

Щелкните заголовок столбца, чтобы отсортировать список по его содержимому. Например, чтобы упростить поиск несанкционированно полученных разрешений на запись, щелкните столбец **Write** и найдите элементы, представляющие группу Все, а также и других пользователей и группы кроме администраторов. Чтобы изменить порядок столбцов, перетащите столбцы на нужное место за заголовок.

Отыскав «сомнительное» разрешение, щелкните соответствующий элемент правой кнопкой мыши, чтобы открыть контекстное меню AccessEnum. Если этот элемент является файлом или папкой, щелчок **Properties** откроет

диалоговое окно **Explorer's Properties (Свойства Проводника)** для данного элемента. Щелкните вкладку **Security** для просмотра или редактирования разрешений на доступ к объекту. Если щелкнуть в контекстном меню **Explore (Проводник)**, откроется окно **Проводника** с данной папкой. Если элемент является разделом реестра, щелчок пункта **Explore** откроет Regedit с этим разделом, позволяя просматривать и редактировать его разрешения. Имейте в виду, что в Windows Vista и выше программа AccessEnum сможет открыть реестр в Regedit, только если она работает на том же IL, что и Regedit.

Можно скрыть один или несколько элементов, щелкнув их правой кнопкой и выбрав **Exclude**. Выбранный элемент и другие элементы, начинающиеся с той же строки, будут скрыты. Например, если исключить C:\Folder, то C:\Folder\Subfolderwill также будет скрыт.

Щелкните кнопку **Save**, чтобы сохранить содержимое списка в текстовый файл в кодировке Unicode с разделителями–знаками табуляции. Выберите **Compare To Saved (Сравнить с сохраненным)** в меню **File**, чтобы отобразить различия по разрешениям между текущим списком и ранее сохраненным файлом. Эту функцию можно использовать для сравнения конфигурации разных компьютеров.

ShareEnum

Такой аспект сетевой безопасности Windows, как защита общих файлов, часто недооценивают. Недостаточная защита постоянно создает проблемы с безопасности, поскольку многие пользователи получают слишком большие права на доступ к файлам на других компьютерах. Если при создании общих файлов в Windows не задавали явно разрешения, по умолчанию группа Все получала полный доступ. В дальнейшем эта группа стала получать только доступ для чтения, но и в этом случае возможен несанкционированный доступ к конфиденциальной информации.

В Windows нет утилит, отображающих все общие ресурсы в сети и их уровень защиты. ShareEnum восполняет этот пробел, отображая все общие файлы и принтеры в домене, диапазоне IP-адресов либо во всей сети для быстрого просмотра в табличном виде (рис. 8-3) и редактирования разрешений на доступ к общим ресурсам.

Поскольку только администратор домена имеет возможность просматривать все сетевые ресурсы, использование ShareEnum наиболее эффективно при запуске из учетной записи администратора домена.

ShareEnum является GUI-программой и не принимает параметры командной строки (кроме **/accepteula)**. Выберите из раскрывающегося списка **All domains (Все домены)**, чтобы проверить всю сеть, **IP address range (Диапазон IP-адресов)**, чтобы проверить диапазон IP-адресов, либо укажите имя домена. Чтобы просканировать часть сети, щелкните **Refresh (Обновить)**. Если выбран пункт **IP address range**, компьютер запросит диапазон IP-адресов для проверки.

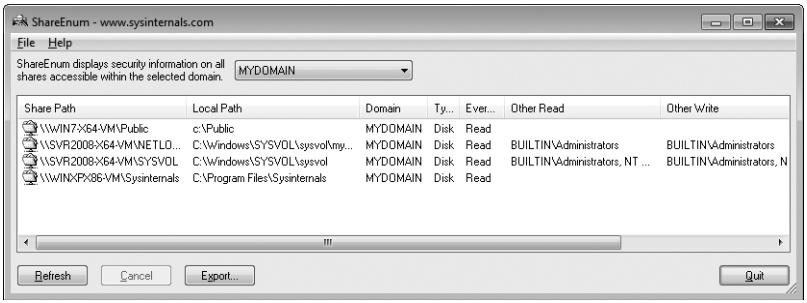


Рис. 8-3. ShareEnum

Щелкните заголовок столбца, чтобы отсортировать список по данным этого столбца. Чтобы изменить порядок столбцов, перетащите заголовки столбцов на нужное место. ShareEnum отображает следующую информацию об общих ресурсах:

- **Share Path (Путь к общему ресурсу)** Имя компьютера и общего ресурса.
- **Local Path (Локальный путь)** Путь к ресурсу в файловой системе удаленного компьютера.
- **Domain** Домен.
- **Type (Тип)** Указывает, является ли ресурс общим файлом (Disk), принтером (Printer) или неизвестным ресурсом (Unknown).
- **Everyone** Разрешения на доступ к ресурсу у группы **Все**, разделенные на категории «чтение», «запись», «чтение/запись», либо пустые поля, если у группы **Все** нет разрешений.
- **Other Read (Чтение для остальных)** Пользователи, кроме членов группы **Все**, которым предоставлено разрешение на чтение общего ресурса.
- **Other Write (Запись для остальных)** Пользователи, кроме членов группы **Все**, которым предоставлено разрешение на изменение или полный доступ к общему ресурсу.
- **Deny** Любые объекты, которым явно запрещен доступ к общему ресурсу.

Щелкните кнопку **Export**, чтобы сохранить содержимое списка в текстовый файл в кодировке Unicode с разделителями-табуляторами.

Выберите **Compare To Saved** в меню **File**, чтобы сравнить разрешения из текущего списка и ранее экспортированного файла.

Для изменения разрешений общего ресурса щелкните его в списке правой кнопкой и выберите **Properties** — ShareEnum откроет окно редактора разрешений на доступ к общему ресурсу. Чтобы открыть общий файл в **Проводнике**, щелкните его правой кнопкой и выберите **Explore**.

ShellRunAs

В Windows XP и Windows Server 2003 можно запускать программу от имени другой учетной записи, щелкнув ее правой кнопкой мыши в **Проводнике,** выбрав команду **Run As (Запуск от имени)** в контекстном меню и введя учетные данные в одноименном окне. Эту функцию часто использовали для запуска программ от имени администратора из-под учетной записи обычного пользователя. Начиная с Windows Vista, эта функция заменена командой **Run As Administrator (Запуск от имени Администратора)**, инициирующей повышение привилегий UAC. Для тех, кто использовал **Запуск от имени**, единственной альтернативой стала менее удобная консольная программа Runas.exe. Для восстановления GUI-команды **Запуск от имени** я написал совместно с Джоном Шварцем, членом команды разработчиков Windows, программу ShellRunAs.

Примечание Некоторые функции ShellRunAs «вернулись» в Windows 7. Если щелкнуть программу или ярлык правой кнопкой, удерживая клавишу Shift, в контекстное меню будет добавлена команда **Run As A Different User**.

ShellRunAs позволяет запускать программу от имени другого пользователя из контекстного меню, запрашивая имя и пароль этого пользователя (рис. 8-4) или PIN-код его смарт-карты в системах, настроенных на вход с помощью смарт-карты. ShellRunAs работает как Runas.exe, но имеет более удобный графический интерфейс. Ни одна из функций ShellRunAs не требует наличия прав администратора и даже не регистрирует элементы контекстного меню. ShellRunAs работает в Windows XP и выше.

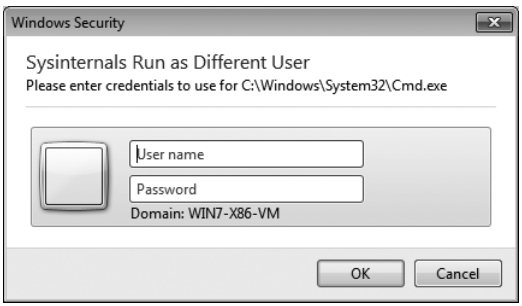


Рис. 8-4. ShellRunAs запрашивает учетные данные

ShellRunAs также поддерживает параметр Runas.exe *netonly*, который не поддерживался ранее графическим интерфейсом пользователя Windows. С параметром *netonly* запускаемая программа использует контекст безопасности запускающего ее пользователя для локального доступа, но для удаленного доступа использует альтернативные учетные данные (рис. 8-5). Имейте в виду, что при запуске программы с параметром *netonly* окно консоли на мгновение «мигает».

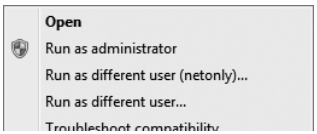


Рис. 8-5. Команда Run As Different User в контекстном меню Проводника

Параметры командной строки для ShellRunAs описаны ниже.

```
ShellRunAs /reg [/quiet] ShellRunAs /regnetonly [/quiet] ShellRunAs /unreg
[/quiet]
```

- **/reg** Регистрирует программу как команду контекстного меню **Проводника** для текущего пользователя (рис. 8-5.).
- **/regnetonly** Регистрирует команду с параметром Netonly в контекстном меню **Проводника** для текущего пользователя.
- **/unreg** Удаляет ShellRunAs из контекстного меню текущего пользователя.
- **/quiet** Не отображает диалоговое окно результатов регистрации или удаления.

```
ShellRunAs [/netonly] программа [аргументы]
```

Этот синтаксис позволяет запускать программы из командной строки ShellRunAs. С помощью параметра **/netonly** можно указать, что введенные учетные данные должны использоваться только для удаленного доступа.

Autologon

С помощью программы Autologon легко настраивать встроенную функцию автоматического входа в Windows под заданной записью пользователя во время загрузки компьютера. Для активации автоматического входа просто запустите Autologon, введите действительные учетные данные в и щелкните кнопку **Enable**. Также можно передать имя пользователя, домен и пароль как аргументы командной строки:

```
autologon Abby MYDOMAIN Pass@word1
```

Пароль хранится в реестре как зашифрованные данные LSA. При следующем запуске системы Windows попытается использовать введенные учетные данные для входа. Имейте в виду, что Autologon не проверяет принимаемые учетные данные, равно как и право учетной записи на вход в систему. Также учтите, что пользователь с правами администратора может без труда восстановить и дешифровать зашифрованные данные LSA, хранимые в реестре.

Для отключения автоматического входа запустите Autologon и щелкните кнопку **Disable** или нажмите клавишу Escape. Для однократного отключения автоматического входа удерживайте клавишу Shift во время загрузки, начиная с того момента, когда должен произойти автоматический вход. Автоматический вход может быть запрещен групповой политикой. Autologon работает в Windows XP и выше и требует привилегий администратора.

LogonSessions

Программа LogonSessions выводит список активных сеансов LSA. Сеанс создается после аутентификации учетной записи пользователя или службы в Windows одним из следующих способов:

- путем интерактивного входа пользователя на консоли или через удаленный рабочий стол;
- через сетевую аутентификацию при доступе к общему файлу или сетевому приложению;
- с помощью SCM, используя сохраненные учетные данные службы;
- с помощью службы Secondary Logon с использованием Runas.exe;
- путем простой проверки операционной системой, как для учетной записи System и NT AUTHORITY\ANONYMOUS LOGON. Данный способ используется при выполнении действий от имени пользователя, не прошедшего аутентификацию или маркера олицетворения уровня «identify».

Маркер доступа создается вместе с сеансом входа и обозначает контекста защиты учетной записи. Маркер доступа копируется процессами и потоками, работающими в данном контексте безопасности, и содержит ссылку на свой сеанс входа. Сеанс остается активным, пока существует хотя бы один экземпляр маркера, ссылающегося на этот сеанс.

Каждый сеанс входа имеет локально уникальный идентификатор (LUID). LUID — это сгенерированная системой 64-разрядная величина, уникальность которой гарантируется в течение сеанса, в котором она сгенерирована. Некоторые LUID являются стандартными. Например, LUID сеанса учетной записи SYSTEM — 0x3e7 (999 в десятичной системе), LUID сеанса Network Service — 0x3e4 (996), а LUID Local Service — 0x3e5 (997). Большинство остальных LUID генерируется случайным образом.

Существует несколько ресурсов, принадлежащих сеансам. К ним относятся сеансы SMB и назначенные буквы сетевых дисков (например, командой NET USE), а также связи Subst.exe. Их можно увидеть в пространстве имен Диспетчера объектов Windows с помощью программы Sysinternals WinObj (см. главу 14) в папке \Sessions\0\DosDevices\LUID. Ресурсы, принадлежащие сеансу SYSTEM, находятся в глобальном пространстве имен.

Имейте в виду, что сеансы LSA не связаны с сеансами служб терминалов (TS). Сеансы TS включают сеансы интерактивных пользователей, вошедших через консоли или удаленный рабочий стол, а также «сеанс 0», в котором работают все служебные процессы. Маркер доступа процесса идентифицирует сеанс LSA, от которого он унаследован, и сеанс TS, в котором он работает. Несмотря на то, что большинство процессов, работающих под учетной записью SYSTEM (сеанс 0x3e7), связано с сеансом 0, два системных процесса работают в каждом интерактивном сеансе TS (Winlogon.exe и Csrss.exe). Их можно увидеть, выбрав столбец **Session** в Process Explorer.

LogonSessions работает в Windows XP и выше, и требует наличия привилегий администратора. Запустите программу LogonSessions в командной

строке с повышенными привилегиями, и она отобразит информацию о всех активных сеансах, включая LUID, имя пользователя и SID учетной записи, прошедшей аутентификацию, использованный пакет аутентификации, тип входа (Service или Interactive), ID сеанса TS, с которым связан сеанс входа, время входа (по местному времени), имя сервера, выполнившего аутентификацию, доменное имя DNS и основное имя пользователя (UPN) учетной записи. Если добавить в командную строку параметр /p, то LogonSessions перечислит под каждым сеансом входа все процессы с маркером, связанным с сеансом входа. Вот пример вывода LogonSessions:

```
[0] Logon session 00000000:000003e7:
   User name:     MYDOMAIN\WIN7-X64-VM$
   Auth package:  Negotiate
   Logon type:    (none)
   Session:       0
   Sid:           S-1-5-18
   Logon time:    6/9/2010 23:02:35
   Logon server:
   DNS Domain:    mydomain.lab
   UPN:           WIN7-X64-VM$@mydomain.lab
[1] Logon session 00000000:0000af1c:
   User name:
   Auth package:  NTLM
   Logon type:    (none)
   Session:       0
   Sid:           (none)
   Logon time:    6/9/2010 23:02:35
   Logon server:
   DNS Domain:
   UPN:
[2] Logon session 00000000:000003e4:
   User name:     MYDOMAIN\WIN7-X64-VM$
   Auth package:  Negotiate
   Logon type:    Service
   Session:       0
   Sid:           S-1-5-20
   Logon time:    6/9/2010 23:02:38
   Logon server:
   DNS Domain:    mydomain.lab
   UPN:           WIN7-X64-VM$@mydomain.lab
[3] Logon session 00000000:000003e5:
   User name:     NT AUTHORITY\LOCAL SERVICE
   Auth package:  Negotiate
   Logon type:    Service
   Session:       0
   Sid:           S-1-5-19
   Logon time:    6/9/2010 23:02:39
```

```
   Logon server:
   DNS Domain:
   UPN:
[4] Logon session 00000000:00030ee4:
   User name:     NT AUTHORITY\ANONYMOUS LOGON
   Auth package:  NTLM
   Logon type:    Network
   Session:       0
   Sid:           S-1-5-7
   Logon time:    6/9/2010 23:03:32
   Logon server:
   DNS Domain:
   UPN:
[5] Logon session 00000000:0006c285:
   User name:     MYDOMAIN\Abby
   Auth package:  Kerberos
   Logon type:    Interactive
   Session:       1
   Sid:           S-1-5-21-124525095-708259637-1543119021-20937
   Logon time:    6/9/2010 23:04:06
   Logon server:
   DNS Domain:    MYDOMAIN.LAB
   UPN:           abby@mydomain.lab
[6] Logon session 00000000:000709d3:
   User name:     MYDOMAIN\Abby
   Auth package:  Kerberos
   Logon type:    Interactive
   Session:       1
   Sid:           S-1-5-21-124525095-708259637-1543119021-20937
   Logon time:    6/9/2010 23:04:06
   Logon server:
   DNS Domain:    MYDOMAIN.LAB
   UPN:           abby@MYDOMAIN.LAB
```

Поскольку учетные записи System и Network Service могут проходить аутентификацию по учетной записи компьютера, их имена выглядят так: **domain\computer$** (или **workgroup\computer$**, если компьютер не присоединен к домену). Сервер входа — имя компьютера для локальных учетных записей, эта строка может быть пустой при входе с кешированными учетными данными.

Также имейте в виду, что в Windows Vista и выше при активном UAC в результате интерактивного входа пользователя, являющегося членом группы Администраторы,[1] создаются два сеанса входа, как видно из предыдуще-

[1] Точнее, два сеанса создаются, когда пользователь входит в стандартную группу с широкими возможностями или обладает привилегиями, эквивалентными административным, например **SeDebugPrivilege.**

го примера с MYDOMAIN\Abby в элементах [5] и [6]. Один из сеансов входа содержит маркер, означающий полные права пользователя, а другой — отфильтрованный маркер с отключенными «мощными» группами и снятыми повышенными привилегиями. Именно по этой причине при повышении привилегий буквы дисков, назначенные в процессах без повышенных привилегий, не действуют в процессах с повышенными привилегиями. Эти и другие данные для всех сеансов можно увидеть в каталоге \Sessions\0\DosDevices\LUID с помощью программы WinObj, описанной в главе 14 (см. также статью 937624 Knowledge Base о настройке EnableLinkedConnections по адресу *http://support.microsoft.com/kb/937624*).

SDelete

Защита от повторного использования объектов является основой политики системы безопасности Windows. Это означает, что приложения, получившит место диске или в виртуальной памяти, yt могут прочитать данные, которые ранее хранились в этой области памяти. Перед тем, как предоставить приложению какой-либо ресурс, Windows выполняет очистку памяти и записывает нули в сектора диска, ранее содержавшие другие данные. Тем не менее, этот механизм не очищает место на диске при удалении файла, поскольку разработчики Windows полагали, что только операционная система будет управлять доступом к системным ресурсам. Но когда операционная система не загружена, с помощью простых редакторов диска и инструментов для восстановления и просмотра данных можно без труда просматривать данные, удаленные операционной системой. Даже если зашифровать файлы с помощью EFS, исходные незашифрованные данные могут оставаться на диске после создания зашифрованной версии файла. Временные файлы также могут не шифроваться.

Единственным способом гарантировать невозможность восстановления удаленных файлов и файлов, зашифрованных EFS, является использование специальной программы. Такие программы перезаписывают данные удаленного файла на диске, используя приведенные ниже технологии, делая невозможным их восстановление. Даже при использовании технологий, способных считывать с магнитных носителей данные файлов, удаленных неделю назад, восстановить данные оказывается невозможно. Одной из таких программ является утилита SDelete (Secure Delete). Ее можно использовать как для гарантированного удаления существующих файлов, так и стирания любых данных в свободных областях диска (удаленных или зашифрованных файлов). SDelete соответствует стандарту удаления и очистки данных DOD 5220.22-M Министерства обороны США, так что можете быть уверены: файлы, стертые с помощью SDelete, исчезнут навсегда. Имейте в виду, что SDelete гарантированно удаляет только данные, но не имена файлов.

Использование SDelete

SDelete — это утилита командной строки, которая работает в Windows XP и выше и не требует наличия прав администратора. Программа использует различный синтаксис для удаления файлов и очистки свободного пространства на диске. Для удаления одного или нескольких файлов или папок используется следующий синтаксис:

```
sdelete [-p число_проходов] [-a] [-s] [-q] файлы
```

где *файлы* — имя файла или папки (поддерживаются знаки подстановки). Параметр -p указывает количество циклов перезаписи каждого файла (по умолчанию задан один проход). Параметр -a необходим для удаления файлов, доступных только для чтения. Параметр -s выполняет рекурсивный обход вложенных папок. Параметр -q («тихий» режим) отключает вывод отчета по каждому файлу. Ниже представлено несколько примеров:

```
REM Securely deletes secret.txt in the current folder
sdelete secret.txt

REM Securely deletes all *.docx files in the current folder and subfolders
sdelete -s *.docx

REM Securely deletes the C:\Users\Bob folder hierarchy
sdelete -s C:\Users\Bob
```

Для очистки свободного места на диске используется следующий синтаксис:

```
sdelete [-p число_проходов] [-z|-c] [d:]
```

Существует два способа перезаписи свободного пространства: параметр -c перезаписывает его случайными данными, а параметр -z — нулями. Параметр -c поддерживает соответствие стандарту Министерства обороны, а параметр -z облегчает сжатие и оптимизацию виртуальных жестких дисков. Параметр -p указывает число проходов. Если буква диска не указана, то очищается свободное пространство текущего тома. Имейте в виду, что после буквы диска должно быть двоеточие.

Примечание Команда **Cipher /W** в Windows аналогична параметру **-c** программы **SDelete**: она заполняет случайными данными все свободное место жесткого диска кроме MFT.

Учтите также, что во время очистки свободного пространства Windows может отображать предупреждение о том, что дисковое пространство заканчивается. Это предупреждение можно игнорировать (о причине его появления — в следующем разделе).

Принцип работы SDelete

Гарантированное удаление файла, не имеющего специальных атрибутов, является относительно несложной операцией: программа безопасного удаления просто перезаписывает данные файла. Сложнее гарантированное удаление сжатых, зашифрованных и разреженных файлов, как и очистка дискового пространства.

Сжатые, зашифрованные и разреженные файлы обрабатываются файловой системой NTFS блоками по 16 кластеров. Если программа выполняет запись в существующий блок такого файла, NTFS выделяет новое пространство на диске для хранения новых данных, а после их освобождает кластеры, которые ранее были заняты файлом. В NTFS столь консервативный подход применяется для обеспечения целостности данных, а также (в случае сжатых и разреженных файлов) в ситуациях, когда выделенное место больше существующего (например, объем новых сжатых данных больше, чем старых). Таким образом, перезапись подобного файла не приводит к удалению его содержимого с диска.

Для обработки таких файлов SDelete использует API-интерфейс дефрагментации. При этом SDelete точно определяет, какие кластеры на диске заняты данными, принадлежащими сжатым, разреженным и зашифрованным файлам. Зная, какие кластеры содержат данные файла, программа SDelete открывает диск для прямого доступа и перезаписывает эти кластеры.

Очистка свободного пространства — тоже непростая задача. Поскольку файловые системы FAT и NTFS не дают приложениям напрямую работать со свободным местом, у SDelete есть два варианта действий. Первый заключается в том, что, как и в случае сжатых, разреженных и зашифрованных файлов, программа может открыть диск для прямого доступа и записать данные в свободном месте. У этого подхода есть существенный недостаток: даже если бы SDelete могла вычислять свободные области на дисках с NTFS и FAT (что весьма непросто), возник бы риск конфликта с текущими файловыми операциями системы. Допустим, SDelete узнала, что кластер свободен, и в этот момент драйвер файловой системы (FAT или NTFS) решил выделить кластер под файл, который изменяет другая программа. Драйвер файловой системы записывает в кластер новые данные, а SDelete перезаписывает только что записанные данные... В результате новые данные файла пропадут. Еще хуже, если этот кластер будет выделен для метаданных файловой системы: в этом случае SDelete повредит структуру файловой системы.

Второй подход, который и применяет SDelete, заключается в опосредованной перезаписи свободного пространства. Сначала SDelete выделяет файл максимально возможного размера с помощью некешируемой операции файлового ввода-вывода, так что полезное содержимое кеша NTFS при этом не пропадает. Поскольку при такой операции размер выделяемого файла округляется по границам сектора (т.е. до 512 байтов), может остаться место, не занятое SDelete, при этом SDelete не сможет увеличить размер своего

файла. Чтобы заполнить оставшееся место, SDelete выделяет (с помощью кешируемой операции) файл максимально возможного размера и перезаписывает содержимое обоих файлов, гарантируя очистку места на диске, до того бывшего свободным.

В случае NTFS-дисков SDelete не всегда ограничивается выделением и перезаписью двух файлов, поскольку программа должна очистить все свободные разделы MFT файлами, помещающимися в записи MFT. Типичный размер записи MFT — 1 Кб, и каждому файлу или каталогу на диске требуется не менее одной записи MFT. Небольшие файлы хранятся внутри своих записей, в то время как файлы, не помещающиеся в отведенное под запись MFT место, распределяются по кластерам за пределами MFT. Все что нужно сделать SDelete со свободным пространством MFT — это записать в нем файл максимально возможного размера. Когда этот файл займет все доступное пространство в записи MFT, NTFS воспрепятствует его дальнейшему росту, поскольку на диске не останется свободных кластеров (они уже заняты двумя файлами, ранее созданными SDelete). После этого SDelete повторяет процесс. Когда создать новый файл будет невозможно, SDelete определяет, что все свободное место MFT гарантированно очищено путем перезаписи.

Чтобы перезаписать имя удаленного файла, SDelete переименовывает файл 26 раз подряд, каждый раз заменяя все символы имени файла очередной буквой алфавита. Например, после первого переименования имя файла *sample.txt* будет выглядеть так: *AAAAAA.AAA*.

SDelete не очищает имена файлов во время очистки свободного места, потому что для этого нужен прямой доступ к структуре каталогов. В структуре каталогов может быть свободное место, содержащее имена удаленных файлов, но в нем нельзя выделять место для файлов. Поэтому SDelete не удастся просто создать в нем файл и перезаписать его.

Глава 9

Утилиты для работы с Active Directory

Компанией Sysinternals созданы три утилиты, упрощающие работу со службой каталогов Active Directory, а также диагностику и устранение ее неполадок:

- **Ad Explorer** — усовершенствованная программа для просмотра и редактирования объектов Active Directory;
- **AdInsight** — программа для мониторинга вызовов API-интерфейсов протокола LDAP в реальном времени;
- **Ad Restore** — средство для перечисления и восстановления случайно удаленных объектов Active Directory.

AD Explorer

Active Directory Explorer (AdExplorer) — это низкоуровневая программа для просмотра и редактирования объектов Active Directory. По своим возможностям AdExplorer почти не уступает программе ADSI Edit из Windows, но, благодаря многофункциональности и простоте использования. Можно использовать AdExplorer для просмотра базы данных Active Directory и атрибутов объектов, не открывая при этом диалоговые окна. Программа также позволяет редактировать свойства, атрибуты и разрешения объектов, а также напрямую переходить от объекта к его схеме, определять избранные расположения, выполнять сложный поиск и сохранять его результаты для последующего использования, делать снимки базы данных Active Directory для просмотра и сравнения в автономном режиме (без подключения к базе данных). Кроме того, AdExplorer автоматически открывает все доступные контексты именования Active Directory, так что вам не придется подключаться по отдельности к конфигурации, схеме и т. д.

Подключение к домену

AdExplorer может отображать в виде дерева сразу несколько доменов и сохраненных ранее снимков. С помощью диалогового окна **Connect To Active Directory (Подключение к Active Directory)**, изображенного на рис. 9-1, можно подключиться к серверу каталогов или открыть сохраненный сни-

мок. Это окно открывается кнопкой **Open** на панели инструментов либо одноименной командой меню **File**. AdExplorer открывает данное окно и при запуске, если прежние подключения не сохранены, а в командную строку не добавлен параметр **-noconnectprompt**.

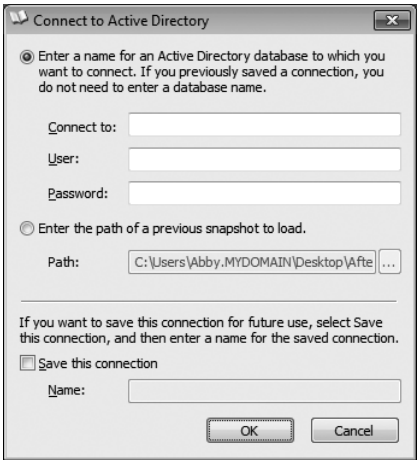


Рис. 9-1. Окно Connect To Active Directory в AdExplorer

К службам каталогов, с которыми работает AdExplorer, относятся Active Directory, Active Directory Lightweight Directory Services (LDS), и Active Directory Application Mode (ADAM). Для подключения к серверу каталогов введите его доменное имя Active Directory, имя либо IP-адрес, имя пользователя и пароль авторизованной учетной записи. Можно подключаться к домену Active Directory по умолчанию, используя данные учетной записи, под которой вы работаете, установив переключатель в первую позицию и оставив текстовые поля пустыми.

Чтобы открыть ранее сохраненный снимок, выберите переключатель в окне во вторую позицию и найдите файл снимка. Имейте в виду, что снимки доступны только для чтения, запрещено также удаление и редактирование объектов, их атрибутов и разрешений. Более подробно о снимках будет рассказано в следующем разделе.

Флажок **Save This Connection (Сохранить подключение)** сохраняет информацию о подключении или снимок, поэтому при следующем запуске AdExplorer он снова подключается к тому же домену или снимку. Имейте в виду, что AdExplorer не сохраняет ваш пароль вместе с подключением к домену по соображениям безопасности, так что вам придется вводить его заново при каждом подключении. Для удаления ранее сохраненного подключения выделите нужное подключение в дереве и выберите пункт **Remove (Удалить)** в меню **File** или в контекстном меню.

Для удаления каталога из окна AdExplorer щелкните правой кнопкой его корневой узел и выберите команду **Remove** в контекстном меню. Можно также удалить подключение, выделив любой объект в дереве и выбрав **Remove** в меню **File**.

Отображение данных AdExplorer

AdExplorer отображает информацию на двух панелях: на левой — дерево объектов, а на правой —перечень атрибутов объекта, выбранного на левой панели. Как показано на рис. 9-2, каждый объект в дереве обозначен именем (например, CN=Abby) и значком из Active Directory. Составное имя объекта (DN) можно получить, выполнив проход вверх по дереву (от объекта к корню) и дописывая к имени объекта имена объектов, встретившихся по пути. DN также отображается в текстовом поле **Path (Путь)** над панелями. Чтобы скопировать DN в буфер обмена, нужно выбрать **Copy Object Name (Копировать имя объекта)** в меню **Edit** или щелкнуть объект правой кнопкой и выбрать соответствующий пункт контекстного меню.

Перечень атрибутов выбранного объекта выводится на правой панели в виде таблицы из четырех столбцов, расположенных в алфавитном порядке. Столбец **Syntax** показывает тип данных атрибута, столбец **Count** — количество значений, которые имеет атрибут (атрибуты могут иметь несколько значений), а столбец **Value(s)** — значение(я) атрибута.

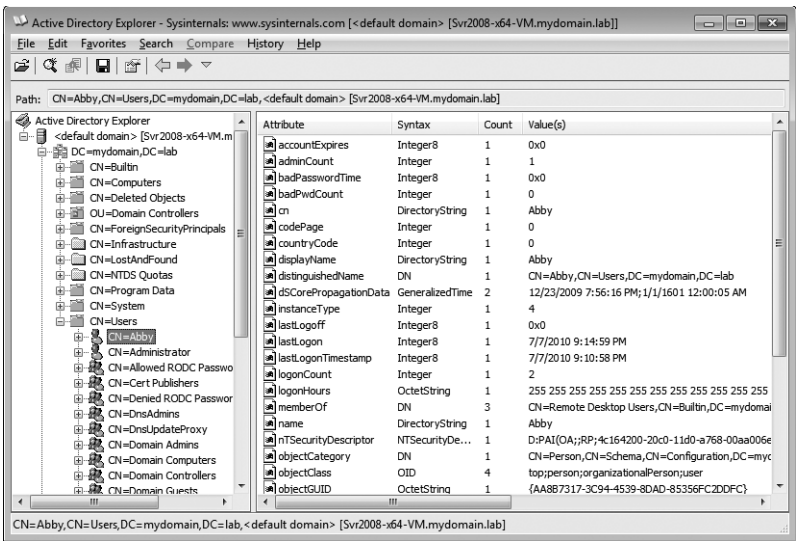


Рис. 9-2. Главное окно AdExplorer

AdExplorer ведет журнал переходов между объектами. Можно пролистывать этот журнал командами **Back (Назад)** и **Forward (Вперед)** из меню **History (Журнал)** или соответствующими кнопками на панели инструментов. Для просмотра всей истории навигации щелкните кнопку **History** на панели инструментов или выберите **History | All (Журнал | Все)**. Чтобы перейти к конкретному объекту журнала, нужно выбрать его в списке.

Чтобы запомнить текущий выбранный объект в иерархии Active Directory, выберите **Add To Favorites (Добавить в Избранное)** в меню **Favorites (Избранное)** и указать имя выбранного элемента. Позднее можно вернуться

к нему, выбрав его в меню **Favorites**. Чтобы переименовать или удалить объект в списке меню **Favorites**, откройте это меню, щелкните правой кнопкой имя объекта и выберите **Rename (Переименовать)** или **Delete (Удалить)**.

Объекты

Для просмотра дополнительной информации об объекте щелкните его правой кнопкой и выберите **Properties (Свойства)** в контекстном меню. Содержимое вкладок окна **Properties** зависит от того, является ли объект корневым узлом соединения, и, если да, является ли он активным подключением либо снимком.

Окно свойств корневого узла содержит вкладки с перечнем основной информации о подключении и статистику схемы, в частности, число классов и свойств. Если узел является узлом RootDSE (корневой узел активного подключения), в диалоговом окне содержится вкладка **RootDSE Attributes** с перечнем данных о сервере каталогов, в частности, *defaultNamingContext* and *configurationNamingContext*. Окно свойств корневого узла сохраненного снимка содержит путь к файлу снимка на момент его захвата, а также описание, сохраненное вместе со снимком.

Окно свойств некорневых объектов содержит три вкладки: **Object Properties**, **Security**, и **Attributes**. Вкладка **Object Properties (Свойства объекта)** отображает имя объекта, DN, класс объекта и схему. Если щелкнуть кнопку **Go To (Перейти)** рядом со схемой, AdExplorer в главном окне выполнит переход к объекту и выделит его в схеме, чем даст возможность проверить или изменить определение схемы для данного объекта. Вкладка **Security (Защита)** представляет собой стандартный редактор разрешений, позволяющий просматривать и изменять разрешения для объекта. Вкладка **Attributes (Атрибуты)** содержит перечень атрибутов объекта с отдельным списком значений.

Объект можно переименовать или удалить. Для этого нужно выделить его, а затем выбрать команду **Rename** или **Delete** из контекстного меню либо из меню **Edit**. Также можно переименовать объект, щелкнув его еще раз после выделения, а затем введя новое имя.

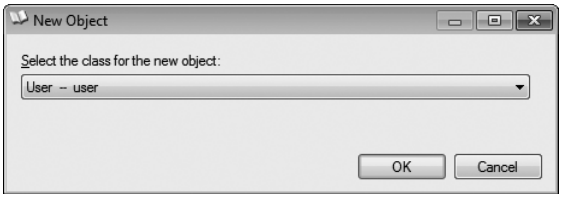


Рис. 9-3. Выбор класса для нового объекта

Чтобы создать новый объект, щелкните родительский контейнер правой кнопкой, выберите из контекстного меню команду **New Object (Новый объект)**, а затем выберите класс для нового объекта из раскрывающегося списка окна **New Object** (рис. 9-3).

После этого AdExplorer откроет окно **New Object — Advanced (Новый объект — Расширенные)**, изображенное на рис. 9-4.

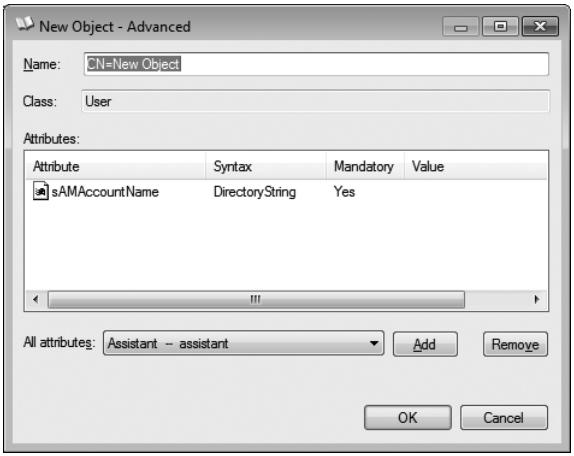


Рис. 9-4. Создание нового объекта: окно New Object — Advanced

Введите имя в текстовом поле **Name** диалогового окна **New Object — Advanced**. Имя должно начинаться с *CN*= и быть уникальным внутри контейнера. Список **Attributes** уже заполнен атрибутами, обязательными для данного класса. Перед созданием объекта их необходимо отредактировать. Для добавления к объекту других атрибутов выберите их из раскрывающегося списка **All Attributes (Все атрибуты)** и щелкните **Add.** Можно удалить добавленный необязательный атрибут, выбрав его в списке и щелкнув **Remove**. Для редактирования атрибута в списке дважды щелкните его — откроется окно **Modify Attribute (Изменить атрибут)**, о котором пойдет речь в следующем разделе.

Атрибуты

При выборе объекта на левой панели главного окна AdExplorer выводит перечень его атрибутов на правой панели. Перечень атрибутов объекта выводится и на вкладке **Attributes** окна свойств объекта. Щелкните правой кнопкой любой атрибут и выберите в контекстном меню **Copy Attributes (Копировать атрибуты),** чтобы скопировать содержимое перечня в буфер в виде значений, разделенных знаком табуляции (также можно выделить атрибут и выбрать **Copy Attributes** в меню **Edit**). Параметр **Display Integers As (Отображать целые числа как)** в этих же меню включает отображение целых чисел в десятичном, шестнадцатеричном формате либо в формате, принятом по умолчанию в AdExplorer.

Чтобы открыть окно свойств атрибутов **Properties** (рис. 9-5), дважды щелкните атрибут или выделите его и выберите **Properties** в меню **Edit**. Окно **Properties** отображает имя атрибута, DN объекта, которому он принадлежит, его синтаксис (тип атрибута), схему и значения. Такое же окно используется

для отображения однозначных и многозначных атрибутов, поэтому значения отображаются в списке, по одному в каждой строке. Щелкните кнопку **Go To** рядом со схемой атрибута, и AdExplorer выполнит переход в каталог, где находится данная схема.

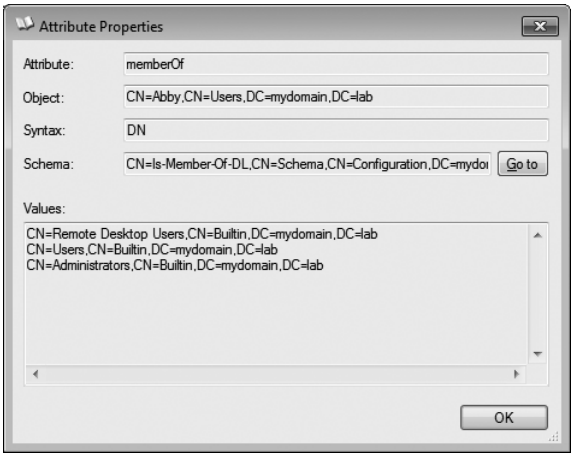


Рис. 9-5. Окно Attribute Properties

Окно **Attribute Properties** доступно только для чтения. Чтобы удалить атрибут, щелкните правой кнопкой атрибут на правой панели и выберите **Delete** в контекстном меню. Для редактирования значения атрибута щелкните атрибут правой кнопкой и выберите **Modify** в контекстном меню. Чтобы установить новый атрибут для объекта, щелкните правой кнопкой любой действующий атрибут и выберите в контекстном меню **New Attribute**. Для добавления нового атрибута или изменения действующего используйте окно **Modify Attribute**, описанное в следующем абзаце. Обратите внимание на то, что удалять, изменять и создавать атрибуты можно также, выделив атрибут и затем выбрав нужный пункт в меню **Edit**.

Окно **Modify Attribute** (рис. 9-6), поддерживает создание и редактирование однозначных и многозначных атрибутов, и выполняет операции с ними одним и тем же способом. Для добавления атрибута к объекту выберите нужный атрибут в раскрывающемся списке **Property**. Для редактирования действующего атрибута выберите его в списке. Новый атрибут не имеет исходного значения. Для ввода нового значения щелкните **Add**. Не добавляйте более одного значения для однозначного атрибута. Можно изменять или удалять действующее значение, выбрав его в списке и щелкнув, соответственно, **Modify** или **Remove**. Имейте в виду, что окно **Modify Attribute** может создавать или изменять одновременно не более одного атрибута. После установки значения (й) атрибута нужно щелкнуть OK, чтобы изменения вступили в силу. Для добавления или редактирования другого атрибута снова выберите, соответственно, **New Attribute** или **Modify**.

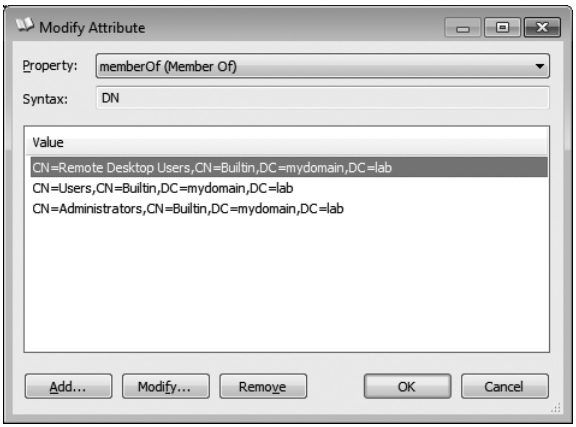


Рис. 9-6. Окно Modify Attribute

Поиск

AdExplorer обладает широкими возможностями для поиска, позволяя находить в контейнере объекты по значениям их атрибутов. Параметры поиска можно сохранять для последующего использования.

Чтобы начать поиск, выберите **Search Container (Поиск в контейнере)** в меню **Search** — откроется окно **Search Container** (рис. 9-7). Для поиска внутри контейнера щелкните контейнер правой кнопкой и выберите пункт **Search Container** в контекстном меню. В этом случае действует критерий *distinguishedName*, ограничивающий поиск выбранным объектом и отходящими от него «ветвями» дерева объектов.

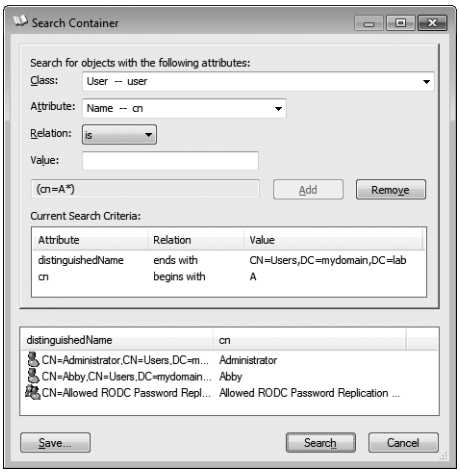


Рис. 9-7. Окно Search Container

Текущие критерии поиска отображаются в списке, расположенном в середине диалогового окна. Для добавления критериев поиска введите в поле со списком **Attribute** искомый атрибут, оператор сравнения и значение, за-

тем щелкните **Add**. Для удаления критерия поиска выберите его в списке и щелкните **Remove**.

Список доступных атрибутов очень велик, поэтому для облегчения поиска выберите в раскрывающемся списке **Class** класс, к которому относится атрибут. После этого в списке останутся только атрибуты, соответствующие схеме данного класса. Атрибуты, имеющие отображаемые имена, отображаются первыми, а перечень остальных находится под заголовком —**Advanced**—. Имейте в виду, что имя класса не используется фильтром, а служит только для облегчения поиска атрибутов в раскрывающемся списке.

Указав критерии поиска, щелкните кнопку **Search** — панель результатов заполнится путями к найденным объектам. Дважды щелкнув объект, можно перейти к нему в главном окне.

Для сохранения критериев поиска щелкните кнопку **Save**. Искомое имя появится в меню **Search**. Можно переименовать или удалить сохраненный результат поиска через контекстное меню, которое открывается щелчком правой кнопкой по сохраненному элементу поиска в меню **Search**.

Снимки

Можно использовать AdExplorer и для сохранения снимков базы данных Active Directory. Впоследствии такие снимки можно открывать с помощью AdExplorer для проверки и поиска объектов и атрибутов Active Directory в автономном режиме. Также можно сравнивать снимки в поисках отличающихся объектов, атрибутов и разрешений. Учтите, что AdExplorer делает снимки только контекстов, конфигурации и схемы именования по умолчанию.

Для сохранения снимка щелкните кнопку **Save** на панели инструментов или выберите **Create Snapshot (Создать снимок)** в меню **File**. Окно **Snapshot** позволяет добавлять комментарии к снимку, указывать место сохранения снимка и замедлять проверку дерева объектов Active Directory для снижения расхода ресурсов контроллера целевого домена.

Загрузив сохраненный снимок (через описанное выше окно **Connect To Active Directory**), можно искать в нем и просматривать его как при подключении к базе данных онлайн. Учтите, что снимки доступны только для чтения, и вносить в них какие-либо изменения нельзя.

После загрузки снимка вы можете сравнить его с другим снимком. Выделите любой объект снимка, а затем выберите **Compare Snapshot** в меню **Compare**, чтобы отобразить окно формирования критериев **Compare Snapshots** (рис. 9-8). Выберите другой снимок, чтобы сравнить его с загруженным снимком. Можно указать классы и атрибуты, по которым будет проводиться сравнение, выбрав их из списков классов и атрибутов. Если вы хотите запомнить класс и атрибут для последующих сравнений, щелкните кнопку **Save** и введите имя — это имя появится в меню **Compare**. Щелкните кнопку **Compare**, чтобы начать сравнение.

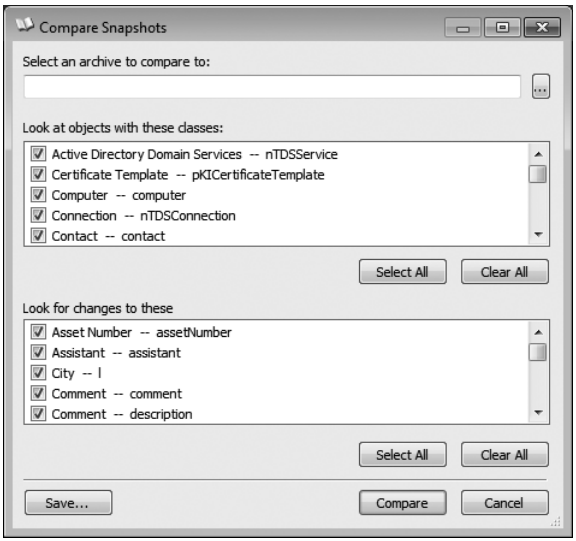


Рис. 9-8. Окно формирования критериев Compare Snapshots

После завершения сравнения выводится перечень различий, как показано на рис. 9-9. При двойном щелчке различия AdExplorer выполняет переход к объекту внутри загруженного снимка. Для изменения сравнения щелкните кнопку **New Compare (Новое сравнение)**, чтобы вернуться в окно формирования критериев.

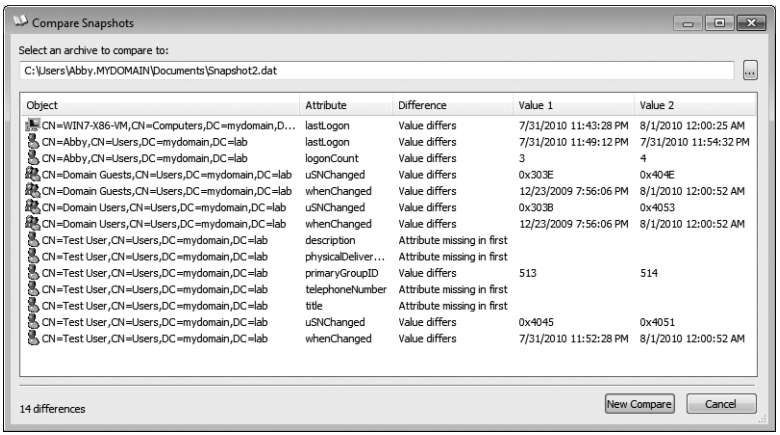


Рис. 9-9. Окно результатов Compare Snapshots

Выберите **Compare Snapshot Security (Сравнить защиту в снимках)** в меню **Compare**, чтобы сравнить разрешения объектов в загруженных снимках. После запуска сравнения дважды щелкните найденное различие — откроется окно **Effective Permissions Comparison (Сравнение действующих разрешений)** с различиями разрешений и полным списком разрешений в снимках 1 и 2.

Можно написать сценарий для AdExplorer, чтобы создавать снимки во время запуска программы, добавив к командной строке параметр **-snapshot**. Для этого нужны два параметра: строка подключения и путь к снимку. *Connection string* — это всего лишь имя сервера. Пустые двойные кавычки задают сервер каталогов по умолчанию. Указать альтернативные учетные данные для соединения невозможно. Для получения снимка домена по умолчанию с помощью текущих учетных данных используйте следующую команду:

```
adexplorer -snapshot «» c:\snapshots\snapshot1.dat
```

Конфигурация AdExplorer

Настройки AdExplorer хранятся в двух разделах реестра. Значение *EulaAccepted* хранится в разделе реестра HKCU\Software\Sysinternals\Active Directory Explorer. Остальные настройки AdExplorer, в том числе, Избранное, пути к снимкам и другие настройки диалогового окна, хранятся в разделе HKCU\Software\MSDART\Active Directory Explorer.

Ad Insight

AdInsight — это программа для мониторинга в режиме реального времени, отслеживающая вызовы LDAP API. Поскольку LDAP является коммуникационным протоколом Active Directory, AdInsight идеально подходит для устранения неполадок клиентских приложений Active Directory.

AdInsight использует технологию «впрыскивания DLL» для перехвата вызовов библиотеки Wldap32.dll, которая является стандартной библиотекой с низкоуровневыми функциями LDAP в Windows. На ней базируются библиотеки более высокого уровня, например, интерфейс ADSI. В отличие от инструментов для мониторинга сети, AdInsight перехватывает и интерпретирует все вызовы клиентских API, включая те, которые не выполняют передачу данных на сервер.

AdInsight отслеживает все процессы, в которые может загрузить свой DLL-файл трассировки. Наиболее надежно программа работает, когда ее запускают в том же контексте безопасности и на том же рабочем столе, что и отслеживаемое приложение. Если клиентское приложение не обладает правами администратора, то AdInsight также не будет обладать ими.

Для отслеживания служб Windows необходимо запустить AdInsight в сеансе 0 служб терминалов. В компьютерах под управлением Windows XP и Windows Server 2003 это выполняется, когда пользователь AdInsight входит в систему через консоль, однако в Windows Vista и выше рабочий стол интерактивного пользователя никогда не находится в сеансе 0. Можно запустить AdInsight в сеансе 0 с помощью следующей команды PsExec с правами администратора:

```
psexec -d -i 0 adinsight.exe
```

В этом случае AdInsight сможет ввести свою трассировочную DLL в другие процессы сеанса 0, включая службы Windows.

Имейте в виду, что DLL, вводимая AdInsight в другие процессы, не выгружаются без риска краха процесса, поэтому введенная DLL остается в памяти процесса до его завершения. Эта DLL обычно не вызывает проблем с хост-процессами, но лучше выполнить перезагрузку после завершения использования AdInsight.

Захват данных AdInsight

Программа AdInsight запускается в режиме захвата и сразу же начинает отслеживать вызовы LDAP API в других процессах и отображать информацию о них в своем главном окне. Как показано на рис. 9-10, верхняя панель AdInsight — панель событий — представляет собой таблицу, где каждая строка обозначает отдельное событие LDAP. Панель дополнительных сведений, расположенная под ней, содержит подробную информацию о параметрах события, выбранного на панели событий. По умолчанию включена автопрокрутка, так что, по мере захвата новых событий, выполняется прокрутка дисплея для их отображения. Можно выключить автопрокрутку с помощью меню **View**, нажав комбинацию клавиш Ctrl+A или щелкнув кнопку **Autoscroll** на панели инструментов. Аналогичным способом можно включать и выключать режим захвата в меню **File** путем нажатия комбинации клавиш Ctrl+E или щелчка кнопки **Capture** на панели управления.

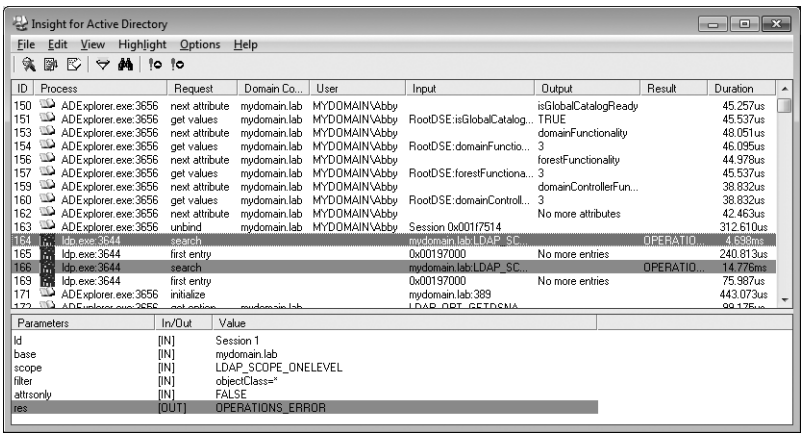


Рис. 9-10. AdInsight

Допускается изменять размеры столбцов на обеих панелях, перетаскивая правую границу заголовка столбца. Также можно перемещать столбцы, перетаскивая их заголовки на новое место. Если данные в столбце не отображаются полностью, наведите указатель мыши на отображаемую часть, и весь текст будет отображен в подсказке.

Можно задать столбцы для отображения, выбрав пункт **Select Columns** в меню **Options** или в контекстном меню заголовка столбца. Выберите в окне

Select Columns (рис. 9-11) столбцы, которые будут отображаться на панели событий и панели дополнительной информации.

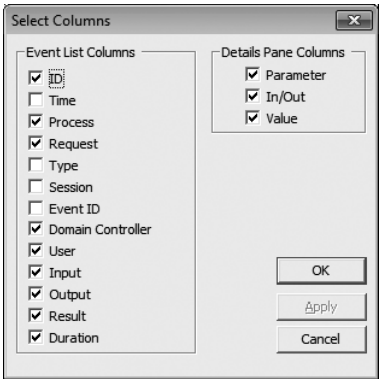


Рис. 9-11. Окно Select Columns

На панели событий могут отображаться следующие столбцы:

- **ID** Уникальный порядковый номер, присвоенный событию программой AdInsight. Пропуски в нумерации могут означать события, удаленные вследствие высокой активности или в результате фильтрации, препятствующей отображению ряда элементов.
- **Time** Время, когда произошло событие. По умолчанию отсчитывается от начала мониторинга программой AdInsight (о других параметрах отображения см. ниже).
- **Process** Имя и PID процесса, выполняющего вызов LDAP, и значок из файла-образа процесса.
- **Request** Имя вызванной функции LDAP. По умолчанию AdInsight отображает простое имя, например: *open, search* или *get values*. Для отображения реального имени функции LDAP, например *ldap_open, ldap_search_s или ldap_get_values* снимите флажок **Show Simple Event Name (Отображать простое имя события)** в меню **Options**.
- **Type** Показывает, является ли запрос синхронным или асинхронным.
- **Session** Описатель сеанса LDAP.
- **Event ID** Описатель события LDAP.
- **Domain Controller** Имя контроллера домена (если есть), к которому был направлен запрос. Если контроллер домена (DC) не был указан, запрос направляется ко всем DC сайта.
- **User** Учетная запись пользователя, используемая для доступа к серверу LDAP. Если контактов с сервером не было, данный столбец остается пустым.
- **Input** Данные, переданные процессом серверу LDAP вместе с запросом. Если передано несколько порций данных, AdInsight отображает в данном столбце одну из них. Все введенные данные, отправленные на сервер, отображаются на панели подробностей.

- **Output** Данные, которые сервер LDAP вернул процессу в ответ на запрос. Если было передано несколько элементов данных, AdInsight отображает один из них. Все данные, полученные от сервера, отображаются на панели подробностей.
- **Result** Код результата, полученный в ответ на запрос. Для упрощения просмотра неудачных операций успешные операции по умолчанию не отображаются. Для отображения успешных операций снимите флажок **Suppress Success Status (Не показывать успешные операции)** в меню **Options**.
- **Duration** Время от начала вызова API до его завершения (см. следующий раздел).

На панели подробностей отображаются параметры ввода-вывода для события, выбранного на панели событий. Доступны следующие столбцы:

- **Parameter** Имена параметров выбранного вызова LDAP.
- **In/Out** Показывает, отправлены ли параметры серверу LDAP («[IN]») или приняты приложением («[OUT]»).
- **Value** Значение параметра, отправляемое или принимаемое процессом.

Для просмотра более подробной информации о запросе щелкните событие правой кнопкой и выберите **Event Information (Информация о событии)**. Откроется всплывающее окно, отображающее имя функции LDAP, краткое описание функции и гиперссылка для поиска дополнительной информации о функции на сайте MSDN Library.

Для просмотра более подробной информации об объекте Active Directory щелкните правой кнопкой событие, связанное с этим объектом, и выберите **Explore**. AdInsight запустит AdExplorer и перейдет к объекту в режиме просмотра AdExplorer.

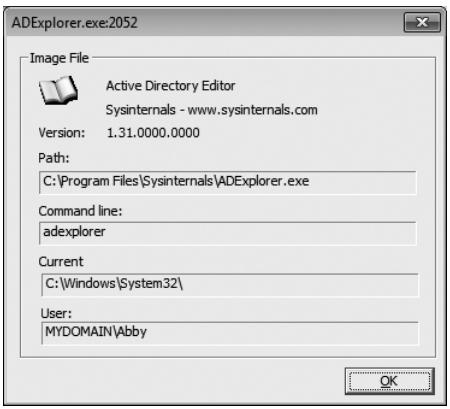


Рис. 9-12. Окно Process Information

Для просмотра более подробной информации о процессе щелкните событие правой кнопкой и выберите **Process Information**. В новом окне (рис. 9-12) откроется информация о процессе, в том числе путь к исполняе-

мому файлу, командная строка, текущий каталоги и учетная запись пользователя, под которой работает процесс.

Для просмотра информации о все процессах, фигурирующих в зарегистрированных данных, выберите команду **View | Processes**. Откроется окно **Processes**, содержащее имя, PID и путь к образу каждого процесса в отчете. Дважды щелкните имя процесса, чтобы открыть окно **Process Information** для данного процесса.

Чтобы очистить панель событий, щелкните кнопку **Clear** на панели инструментов или нажмите клавиши Ctrl+X. Эта операция обнуляет порядковый номер и восстанавливает значения в столбце **Time**, если выбрано относительное время.

По умолчанию AdInsight сохраняет последние 50000 событий и удаляет более старые. Для настройки охвата журнала по времени команду выберите **History Depth** в меню **View** и укажите другое число. Если задано значение 0, AdInsight будет сохранять все события, не удаляя старые. Имейте в виду, что выключение автопрокрутки снимает ограничение по сроку давности событий, и новые события не «вытесняют» старые из списка на экране.

Параметры отображения данных

Помимо изменения шрифта, используемого AdInsight, и размещения окна программы поверх остальных окон (оба параметра находятся в меню **Options**), можно выбрать использование в интерфейсе «простых и понятных» или технических терминов и настраивать их формат.

Параметры отображения времени

По умолчанию столбец **Time** отображает временной промежуток с начала мониторинга программой AdInsight (сбрасывается при командой **Clear Display**). Для вывода местного времени возникновения события выберите **Options | Clock Time**. После включения **Clock Time** меню **Options** также позволяет отображать время с точностью до миллисекунд (параметр **Show Milliseconds**).

В столбцах **Time** (если не активен параметр **Clock Time**) и **Duration** значения отображаются в простом формате, т.е. как число секунд, милли- или микросекунд. Таким образом, слева от разделителя разрядов всегда выводится от одной до трех цифр. Если снять флажок **Show Simple Time (Отображать «простое» время)** в меню **Options**, эти значения будут отображаться в секундах с точностью до восьмого знака после запятой (точки). Например, длительность может обозначаться так: 25.265 мс (т.н. «простое время») или так: 0.025265 (с).

Отображение имен

По умолчанию AdInsight отображает простые имена функций LDAP, например *open, search* или *get values*. Чтобы отображались реальные имена функций LDAP, например *ldap_open, ldap_search_s* или *ldap_get_values,* снимите флажок **Show Simple Event Name** в меню **Options**.

AdInsight отображает составные имена в удобном для восприятия формате, например *mydomain.lab\Users\Abby.* Для просмотра реальных составных имен (например, *CN=Abby,CN=Users,DC= mydomain,DC=lab),* выберите **Show Distinguished Name Format (Показывать составные имена)** в меню **Options**.

Программа AdInsight отображает фильтры LDAP на панели подробностей в понятном формате, например:

```
(( NOT((showInAdvancedViewOnly=TRUE)) AND (samAccountType=805306368)) AND
((name=rchase-2k8*) OR (sAMAccountName=rchase-2k8*)))
```

Если вы предпочитаете стандартный (префиксный) синтаксис LDAP, снимите флажок **Show Simple LDAP Filters (Отображать простые фильтры LDAP)** в меню **Options**. Так выглядит показанный выше фильтр запроса в стандартном формате:

```
(&(&(!(showInAdvancedViewOnly=TRUE))(samAccountType=805306368))
(|(name=rchase-2k8*) (sAMAccountName=rchase-2k8*)))
```

Поиск информации

AdInsight поддерживает несколько средств поиска. К ним относятся поиск текста, визуальная подсветка и навигация.

Поиск текста

Для поиска текста на панели событий нажмите клавиши Ctrl+F или щелкните значок **Find** на панели инструментов — откроется окно **Find** (рис. 9-13). Помимо стандартных параметров (поиск слова целиком, поиск с учетом регистра и направления), окно **Find** позволяет задавать столбцы для поиска. Если искомый текст обнаруживается на панели событий, выводится содержащее его событие, автопрокрутка при этом отключается, чтобы это событие не ушло из окна.

Рис. 9-13. Окно Process

Окно **Find** — не модальное, поэтому возможно переключаться обратно в главное окно AdInsight, не закрывая окно **Find**. Закончив поиск и вернувшись в главное окно, можно повторить поиск в другом направлении (вниз по списку событий — нажатием F3, вверх — нажатием Shift+F3).

Подсветка событий

Подсветка визуально выделяет нужную информацию. По умолчанию события-ошибки подсвечиваются красным, а события длительностью более 50 мс — темно-синим. Для включения или выключения подсветки выберите пункт **Enable Highlighting (Включить подсветку)**, а для настройки подсветки —**Highlight Preferences (Свойства подсветки)** в меню **Highlight** — откроется окно **Highlight Preferences** (рис. 9-14).

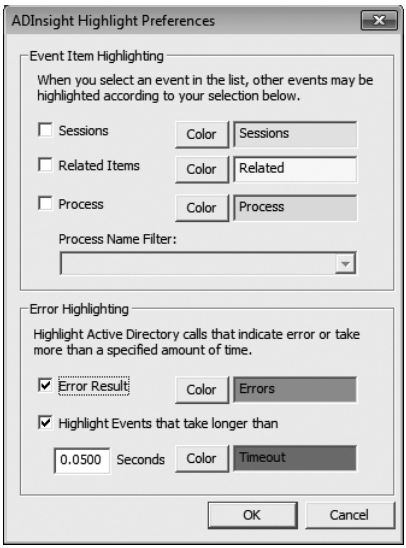


Рис. 9-14. Окно Highlight Preferences

В группе **Event Item Highlighting (Подсветка элементов событий)** пункты **Sessions (Сеансы)** и **Related Items (Связанные элементы)** подсвечивают элементы, сходные с выбранным событием. При выборе другого элемента на панели событий подсвечиваются другие события в соответствии с выбранным элементом. Если выбран параметр **Sessions**, все события с тем же описателем сеанса, что и у выбранного события, подсвечиваются цветом, заданным для этого типа событий (по умолчанию используется черный текст на голубом фоне). Если выбран параметр **Related Items**, подсвечиваются все события с одинаковым описателем события (по умолчанию используется черный текст на желтом фоне).

Для подсветки событий, принадлежащих процессу, выберите **Process** и введите текст, совпадающий с именем (именами) одного или нескольких процессов из списка **Process Name Filter (Фильтр имен процессов)**. События, сгенерированные заданными процессами, будут выделены (по

умолчанию — черным текстом на зеленом фоне). Текст в поле **Process Name Filter** списке обрабатывается как фильтр, т. е. для подсветки *ldp.exe* и *svchost.exe* можно вести **ldp;svchost.**

Группа **Error Highlighting (Подсветка ошибок)** идентифицирует события-ошибки и события, обработка которых длиться дольше запланированного. Эти виды подсветки включаются по отдельности, для них также можно задавать пороговые значения, определяющие подсветку. Имейте в виду, что функция перехода к тексту или к предыдущей ошибке требует включения подсветки результатов ошибок (**Error Result)**.

Для изменения цвета подсветки щелкните кнопку **Color** — откроется окно **Highlight Color**, в котором можно устанавливать цвета текста и фона.

Просмотр связанных событий

AdInsight позволяет открывать окна с перечнем событий, связанных с выбранным событием. Щелкните нужное событие в главном окне AdInsight и выберите команду **View Related Events (Просмотр связанных событий)** либо **View Session Events (Просмотр событий сеанса)** в меню **View** или через контекстное меню.

При выборе **View Related Events** открывается окно **Related Transaction Events (Связанные события транзакций)**. В нем отображается перечень событий из главного окна с тем же описателем, что и у выбранного события. При выборе **View Session Events** открывается окно **Related Session Events**. В нем отображается перечень событий из главного окна с тем же описателем сеанса LDAP, что и у выбранного события.

Окна **Related Events** очень похожи на главное окно AdInsight: и то и другое содержат панель событий и панель дополнительной информации. Настройки их столбцов те же, что и в главном окне, можно изменять размеры и порядок столбцов, однако набор отображаемых столбцов изменить нельзя.

Поиск событий-ошибок

Щелкните кнопку **Goto Next Event Error (Перейти к следующей ошибке)** на панели инструментов — будет выбрано следующее событие на панели событий, которое вернуло ошибочный результат. Чтобы найти и выбрать предыдущую ошибку, щелкните кнопку **Goto Previous Event Error (Перейти к предыдущей ошибке)** на панели инструментов. То же самое можно сделать, щелкнув событие правой кнопкой и выбрав в контекстном меню **Next Event Error** или **Previous Event Error**.

Обратите внимание на то, что эти кнопки на панели инструментов и элементы контекстного меню активны, только если включена подсветка и задан тип подсветки **Error Result**.

Фильтрация результатов

Чтобы уменьшить объем информации, подлежащей анализу, можно настроить фильтры для сбора данных. Фильтры позволяет отбирать отображаемые

события по именам процессов или функций LDAP. Имейте в виду, что фильтры действуют только во время захвата данных, изменение фильтра не влияет на список уже зарегистрированных событий.

Чтобы настроить фильтр для захвата данных, щелкните кнопку **Filter** на панели инструментов или выберите **Event Filter (Фильтр событий)** в меню **View** — откроется окно **Event Filters** (рис. 9-15). Группа **Process Filter (Фильтр процессов)** позволяет задавать имена процессов, чтобы отображать или скрыть соответствующие события. По умолчанию включены все процессы: в строку фильтра **Include** внесена звездочка (*), а фильтр **Exclude** пуст. В полях **Include** и **Exclude** можно указывать несколько строк, разделенных точкой с запятой. Если сгенерированное процессом событие содержит одну или несколько строк, заданных в поле **Exclude**, событие будет скрыто, если же событие совпадает с фильтром **Include** (а с фильтром по умолчанию, *, совпадает любой текст), событие будет отображаться. Если же в поле фильтра **Include** внесены другие строки, событие будет отображаться, только если имя процесса совпадет хотя бы с одной из этих строк. Фильтрация выполняется с учетом регистра символов. Не включайте в фильтры пробелы, если только они не входят в искомый текст.

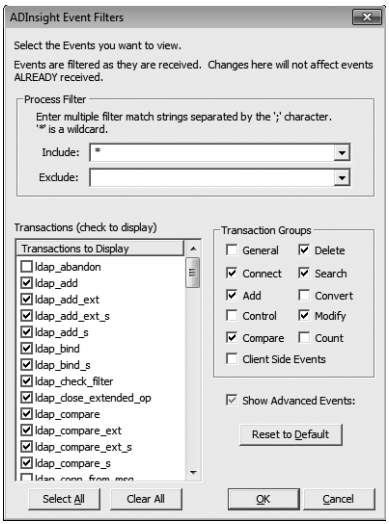


Рис. 9-15. Окно Event Filters

Список **Transactions (Транзакции)** в левой нижней части окна **Event Filters** указывает, какие функции LDAP (транзакции) будут отображаться на панели событий AdInsight. Имейте в виду, что фильтр по умолчанию выделяет не все события. В этом списке можно выделить или снять выделение с отдельных низкоуровневых функций, заданных по имени. Для выделения или очистки списка щелкните кнопку **Select All (Выделить все)** или **Clear All (Очистить все)**. Чтобы одновременно выделить или снять выделение со всех заданных API-функций, установите или снимите соответствующие

флажки в группе **Transaction Groups (Группы транзакций)**. Так, для просмотра только функций, связанных с подключением, привязкой и отключением от сервера щелкните кнопку **Clear All**, а затем установите флажок **Connect**. Для отображения событий, которые обычно не используются для устранения неполадок и настройки, выберите **Show Advanced Events (Показать расширенные события)**.

Для восстановления настроек фильтров по умолчанию щелкните кнопку **Reset To Default (Восстановить значения по умолчанию)**. Учтите следующее: если во время предыдущего сеанса работы AdInsight действовал фильтр процессов, при новом запуске открывается окно **Event Filters** для подтверждения настроек фильтра. Чтобы подавить запрос подтверждения, запускайте программу с параметром -q.

Сохранение и экспорт данных AdInsight

Для сохранения всех данных, захваченных AdInsight, выберите команду **Save** или **Save As** в меню **File**. По умолчанию «родные» файлы AdInsight имеют расширение .wit. В этом формате записываются все зарегистрированные данные с сохранением точности, что позволяет открывать такие файлы в AdInsight на том же или на другом компьютере. Чтобы открыть ранее сохраненный файл AdInsight, нажмите клавиши Ctrl+O или выберите **Open** в меню **File**.

Для сохранения данных AdInsight в текстовом файле нажмите Ctrl+Alt+S или выберите **Export To Text File (Экспортировать в текстовый файл)** в меню **File**. AdInsight экспортирует данные как текстовый файл формата ANSI с разделителями-табуляторами и заголовками столбцов, в котором каждая строка соответствует одному событию. AdInsight выводит окно, в котором можно выбрать экспорт всех данных либо только отображаемых столбцов. Параметр **Include Detailed Information (Включить подробную информацию)** добавляет содержимое панели дополнительной информации к сохраняемым данным в виде дополнительных полей с разделителями-табуляторами. Имейте в виду, что заголовок имеется только у первого дополнительного столбца.

Чтобы скопировать данные с панели событий или панели дополнительной информации в буфер обмена Windows в виде текста с разделителями-табуляторами, выберите строку и нажмите Ctrl+C.

AdInsight можно использовать и для просмотра отчетов о зарегистрированных событиях в браузере (в формате HTML). Выберите **HTML Reports (Отчеты в формате HTML)** в меню **View,** а затем — вид отчета:

- **Events (События)** В этом HTML-отчете содержатся отображаемыt столбцы панели событий, по одному событию в строке. В столбце **Request (Запрос)** отображается гиперссылка на документацию по соответствующей функции на веб-сайте MSDN Library. Имейте в виду, что при наличии большого объема данных этот отчет оказывается очень большим, и загрузка отчета в браузер занимает много времени.

- **Events with Details (События с подробностями)** Этот отчет содержит ту же информацию, что и предыдущий, но к каждой строке добавляется таблица с содержимым панели подробностей для данного события.
- **Event Time Results (Хронометраж событий)** Гистограмма длительности вызовов LDAP на панели событий, число вызовов каждой из функций, общее время всех вызовов, наибольшая длительность и среднее время вызова каждой из функций. Чтобы включить в отчет все функции LDAP, включая те, что не вызвались и не регистрировались программой AdInsight, выберите пункт **Preferences (Настройки)** в меню **Options** и снимите флажок **Suppress Uncalled Functions In Reports (Не показывать невостребованные функции в отчетах)**.
- **Highlighted Events (События с подсветкой)** Аналогичен отчету **Events With Details**, но включает только подсвеченные в данный момент события.

AdInsight создает эти отчеты в папке TEMP. Сохранить их в другом месте можно командой браузера **Save As** либо через Проводник (путь к папке TEMP см. в адресной строке браузера).

Параметры командной строки

В пакетных файлах файлах, сценариях и окне консоли AdInsight можно запускать со следующими параметрами:

```
adinsight [-fi IncludeFilter][-fe ExcludeFilter][-f SavedFile][-q][-o][-t]
```

Вот значения этих параметров:

- -fi *IndudeFilter* Задает имя процесса для фильтра **Include** (подробнее см. выше).
- -fe *ExdudeFilter* Задает имя процесса для фильтра **Exclude** (подробнее см. выше).
- -f *SavedFile* Открывает сохраненный файл AdInsight для просмотра.
- -q Запускает AdInsight, не открывая окно **Filter** (по умолчанию оно отображается при запуске программы, если действуют какие-либо фильтры процессов).
- -o Выключает захват событий после запуска.
- -t Отображает уведомление на панели задач.

Ad Restore

Active Directory в Windows Server 2003 поддерживает средство восстановления удаленных (вернее, *помеченных* для удаления, *tombstoned*) объектов. AdRestore — простая утилита командной строки, которая перечисляет удаленные объекты в домене и дает возможность их восстановления.

Ниже представлен синтаксис командной строки AdRestore:

```
adrestore [-r] [фильтр]
```

При отсутствии каких-либо параметров командной строки AdRestore перечисляет удаленные объекты в текущем домене, отображая CN, DN и последний известный родительский контейнер для каждого объекта. При наличии параметра -r AdRestore отображает объекты поочередно, запрашивая подтверждение на восстановление каждого.

Если задать текст в качестве фильтра, будут отображаться только те объекты, у которых CN совпадает с заданным текстом. Фильтр применяется с учетом регистра, фильтр с пробелами должен заключаться в кавычки. Вот пример поиска удаленных объектов с текстом «Test User» в CN и запросом их восстановления:

```
adrestore -r «Test User»
```

По умолчанию только администраторы домена могут получать список или восстанавливать удаленные объекты, но они могут делегировать эти права и другим. Если у вас нет разрешения на перечисление удаленных объектов, Active Directory (а, следовательно, и AdRestore) просто выводит пустой список, не сообщая об ошибке. Кроме того, при восстановлении удаленных объектов действуют следующие ограничения:

- Помеченный для удаления объект сохраняет только часть исходных атрибутов, поэтому AdRestore не может полностью восстановить его. Для восстановленного объекта пользователя требуется повторная установка пароля.
- Объект не может быть восстановлен, если метка удаления просрочена и объект удален безвозвратно.
- Восстановление объектов из корня контекста именования, например, домены или разделы приложений, невозможно.
- Невозможно восстановление объектов схемы. Никогда не удаляйте их, чтобы не получить недействительных объектов Active Directory.
- Объект не удается восстановить, если его родительский контейнер удален и не восстановлен.
- Восстановление удаленных контейнеров возможно, но восстановить объекты в них сложно, так как иерархию в контейнере приходится воссоздавать вручную.

Глава 10

Утилиты рабочего стола

В отличие от большинства программ компании Sysinternals, утилиты, о которых пойдет речь в данной главе, не предназначены для диагностики и устранения неполадок. BgInfo отображает информацию о конфигурации компьютера в виде фонового рисунка рабочего стола. С помощью Desktops можно запускать приложения на отдельных виртуальных рабочих столах и переключаться с одного рабочего стола на другой. Программа ZoomIt, которую я всегда использую для подготовки презентаций, увеличивает размеры экрана и включает аннотации.

BgInfo

Вспомните, сколько вам приходилось вводить консольных команд или просматривать окон диагностических программ для определения таких важных аспектов конфигурации ПК, как имя компьютера, его IP-адрес или версия оперативной системы.

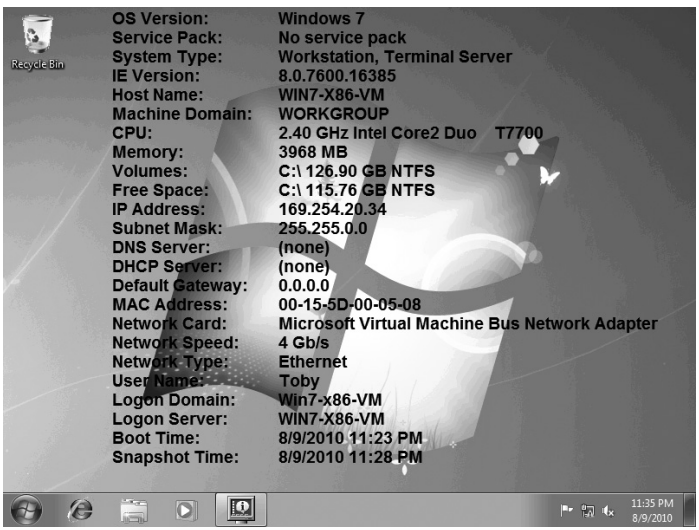


Рис. 10-1. Фоновый рисунок рабочего стола, созданный BgInfo

Утилита BgInfo от Sysinternals может автоматически отображать эти и многие другие данные на фоновом рисунке рабочего стола. BgInfo всегда выводит в удобном для просмотра виде актуальную информацию, полученную в момент вашего входа в систему (рис. 10-1). BgInfo не только отображает большое количество информации, но и поддерживает разнообразные варианты настройки ее вида. Утилита BgInfo завершается после генерации фонового рисунка, поэтому она не потребляет ресурсы системы и не мешает другим приложениям.

При запуске программы BgInfo без параметров командной строки она отображает в правой верхней части диалогового окна редактор конфигурации и начинает 10-секундный времени (Time Remaining), как показано на рис. 10-2, который можно остановить щелчком внутри окна. Когда время таймера истекает, BgInfo устанавливает фоновый рисунок в соответствии с отображаемой конфигурацией, а затем завершает работу.

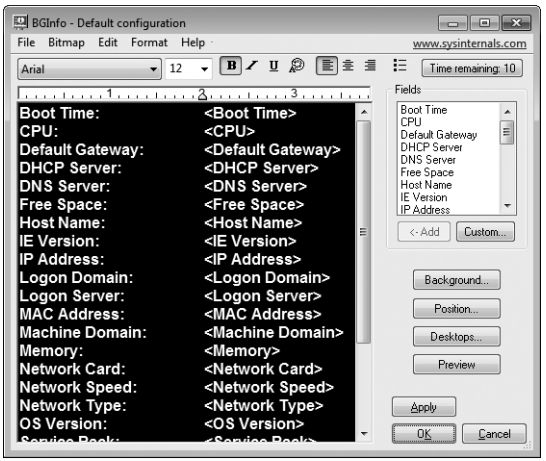


Рис. 10-2. Окно редактора BgInfo с таймером

Настройка данных для отображения

С помощью редактора BgInfo можно настраивать расположение и вид данных для отображения на фоновом рисунке, объединяя текст и (в угловых скобках) поля данных. Конфигурация BgInfo по умолчанию отображает перечень меток и полей данных для всех встроенных полей в алфавитном порядке. Например, когда показанная на рис.10-2 конфигурация используется для генерации фонового рисунка, на нем отображается надпись «Boot Time:» («Время загрузки»), а справа вместо <Boot Time> будет отображаться фактическое время загрузки компьютера.

Чтобы изменить отображаемые поля, просто измените текст в окне редактора. Например, чтобы на рис. 10-2 первой отображалась информация о ЦП, выделите всю строку с надписью «CPU» в окне редактора, нажмите Ctrl+X, чтобы вырезать ее, переместите курсор в верхнюю строку и на-

жмите Ctrl+V. Также можно вставить комментарий и соответствующее поле (в угловых скобках) в текущей позиции курсора в окне редактора, выделив нужный элемент в списке **Fields (Поля)** и щелкнув кнопку **Add** или же просто дважды щелкнув элемент в списке.

Метки не являются обязательными. Например, для того чтобы показать имя вошедшего пользователя в формате DOMAIN\USER, нужно добавить два поля: **<Logon Domain>\<User Name>.** Перечень полей BgInfo см. в табл. 10-1.

Табл. 10-1. Поля данных BgInfo

Название поля	Описание
Атрибуты операционной системы	
OS Version (Версия ОС)	Название операционной системы, например, Windows 7. Если BgInfo не распознает операционную систему, вместо нее она отображает номер версии Windows
Service Pack (Пакет обновления)	Номер пакета обновлений, например, Service Pack 1, или No Service Pack
System Type (Тип системы)	Тип системы, например, рабочая станция (Workstation) или контроллер домена (Domain Controller). В Microsoft Windows XP и выше BgInfo выводит также надпись «Terminal Server» («Сервер терминалов»)
IE Version (Версия IE)	Версия Internet Explorer, как указано в поле Version раздела реестра HKLM\ Software\Microsoft\lnternet Explorer
Host Name (Имя хоста)	Имя компьютера
Machine Domain (Машинный домен)	Домен или рабочая группа, к которой принадлежит компьютер
Атрибуты оборудования	
CPU (ЦП)	Тип ЦП, например, Dual 2.50 GHz Intel Core2 Duo T9300
Memory (Память)	Объем физической оперативной памяти, доступной для просмотра ОС Windows
Volumes (Тома)	Выводит перечень стандартных томов по буквам дисков, отображая общий объем и файловую систему каждого из них
Free Space (Свободное пространство)	Выводит перечень стандартных томов по буквам дисков, отображая свободное пространство и файловую систему каждого из них
Атрибуты сети	
IP Address (IP-адрес)	Выводит перечень IP-адресов для каждого сетевого интерфейса на компьютере
Subnet Mask (Маска подсети)	Выводит перечень масок подсети, связанных с IP-адресами, отображаемыми в предыдущем поле

Табл. 10-1. (*окончание*)

Название поля	Описание
DNS Server (Сервер DNS)	Отображает сервер (серверы) DNS для каждого сетевого интерфейса на компьютере
DHCP Server (Сервер DHCP)	Отображает сервер DHCP для каждого сетевого интерфейса на компьютере
Default Gateway (Шлюз по умолчанию)	Отображает основной шлюз для каждого сетевого интерфейса на компьютере
MAC Address (MAC-адрес)	Отображает MAC-адрес для каждого сетевого интерфейса на компьютере
Network Card (Сетевая карта)	Определяет имя сетевой платы для каждого сетевого интерфейса на компьютере
Network Speed (Скорость сети)	Отображает скорость сети для каждой сетевой платы, например, 100 Mb/s
Network Type (Тип сети)	Отображает тип сети для каждой сетевой платы, например, Ethernet
Атрибуты входа в систему	
User Name (Имя пользователя)	Имя учетной записи пользователя, запускающего BgInfo
Logon Domain (Домен входа)	Домен учетной записи пользователя, запускающего BgInfo
Logon Server (Сервер входа)	Имя сервера, выполнившего аутентификацию пользователя, запускающего BgInfo
Временные метки	
Boot Time (Время загрузки)	Дата и время последней загрузки компьютера
Snapshot Time (Время снимка)	Дата и время создания фонового рисунка BgInfo

Помимо встроенных полей BgInfo, можно добавлять собственные элементы в список **Fields (Поля)**, а затем вставлять их в конфигурацию фонового рисунка. BgInfo располагает целым рядом источников информации, суммированных в табл. 10-2.

Для ввода и управления пользовательскими полями щелкните кнопку **Custom (Пользовательские)** — откроется диалоговое окно **User Defined Fields (Пользовательские поля)**. На рис. 10-3 изображено диалоговое окно с примерами пользовательских полей **Num CPUs** (отображает переменную окружения NUMBER_OF_PROCESSORS) и **Legal Notice Text**, которое отображает текст соглашения из реестра, отображаемый перед входом пользователя в систему, а также версию BIOS, выводимую запросом WMI.

Табл. 10-2. Источники информации BgInfo

Имя поля	Описание
Пользовательские поля	
Environment variable (Переменная окружения)	Параметр переменной окружения
Registry value (Параметр реестра)	Текстовое значение любого параметра реестра
WMI query (Запрос WMI)	Текстовый вывод запроса WMI
File version (Версия файла)	Версия файла
File timestamp (Временная метка файла)	Дата и время последнего изменения файла
File content (Содержимое файла)	Текстовое содержимое файла
VBScript file	Текстовый вывод сценария на VBScript

Чтобы определить новое пользовательское поле, щелкните кнопку **New**. Выберите в списке имеющееся пользовательское поле и щелкните кнопку **Edit** или **Remove**, чтобы изменить или удалить его. Когда вы щелкните **OK**, список **Fields** в главном окне BgInfo обновится.

На рис. 10-4 изображено окно **Define New Field (Определить новое поле)**, которое используется для создания и редактирование пользовательских полей. BgInfo использует в качестве метки по умолчанию введенный идентификатор или имя поля данных. Например, у поля **Num CPUs** будет идентификатор *<Num CPUs>*. Идентификаторы могут содержать только буквы, цифры, пробелы и символ подчеркивания.

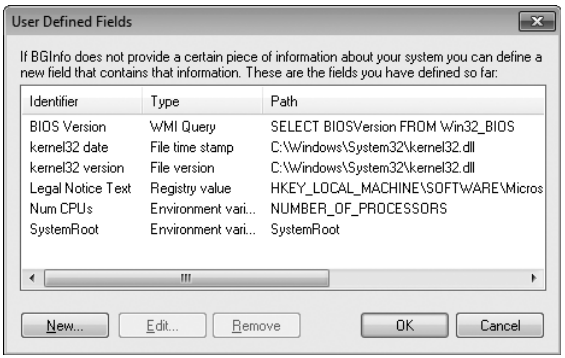


Рис. 10-3. Управление пользовательскими полями

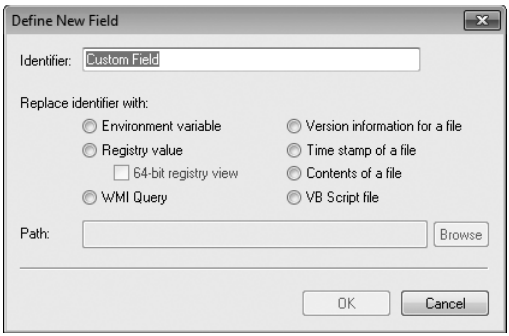


Рис. 10-4. Определение нового пользовательского поля

Выберите тип источника информации и введите его имя в поле **Path**. Кнопка **Browse** откроет окно, тип которого зависит от типа источника. Если выбрать параметр **Environment**, то щелчок кнопки **Browse** откроет список переменных окружения. При выборе параметра **WMI Query** щелчок кнопки **Browse** открывает редактор запросов WMI. При работе с файловыми источниками щелчок кнопки **Browse** открывает стандартное окно выбора файлов. Единственным типом источников, с которым кнопка **Browse** не работает — поле **Registry Value**. В данном случае нужно ввести полный путь к параметру реестра, например:

```
HKLM\Software\Microsoft\Windows NT\CurrentVersion\CurrentBuildNumber
```

При работе в 64-разрядной системе выбор просмотра 64-разрядного реестра **(64-Bit Registry View)** гарантирует, что указанный путь не будет перенаправлен в подраздел *Wow6432Node*.

Параметры внешнего вида

Редактор BgInfo представляет собой текстовый редактор с поддержкой функций отмены и повтора ввода. Можно выделить часть текста или весь текст, изменить начертание шрифта, размер, стиль, выравнивание и маркировку с помощью панели инструментов меню **Format**. Формат текста, сохраняется при вставке из буфера обмена. Перетаскивание маркера на горизонтальной линейке изменяет первую позицию табуляции для выравнивания выбранных абзацев. Можно также добавить к тексту растровое изображение, выбрав **Insert Image (Вставить изображение)** в меню **Edit**.

Щелкните кнопку **Background (Фон)** для выбора фона для рисунка. Как показано на рис. 10-5, BgInfo может наложить данные на текущие обои. Можно указать фоновое изображение и его расположение (по центру, мозаичное или растянутое) или выбрать фоновый цвет. Кнопки **NT 4.0, 2000** и **XP** используются для настройки цветов фона по умолчанию для соответствующих версий Windows. Если установить флажок **Make Wallpaper Visible Behind Text (Отобразить фоновый рисунок позади текста)**, BgInfo наложит текст прямо на изображение обоев. Если снять этот флажок, BgInfo вставит текст внутри прямоугольника, залитого выбранным цветом, поверх обоев.

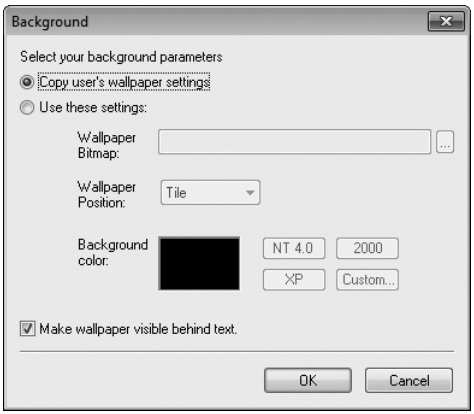


Рис. 10-5. Окно BgInfo Background

Щелкните кнопку **Position (Расположение)**, чтобы указать место размещения текста на экране. Выберите вариант расположения из списка **Locate On Screen (Разместить на экране)** из окна **Set Position (Установить расположение)** (рис. 10-6). К длинному тексту (такому, как имена сетевых плат) можно применить параметр **Limit Lines To (Ограничить строки до)**. Выбор параметра **Compensate For Taskbar Position (Ввести поправку на расположение панели задач)** гарантирует, что панель задач не будет закрывать текст. Если к компьютеру подключено несколько мониторов, щелкните кнопку **Multiple Monitor Configuration (Конфигурация для нескольких мониторов)**, чтобы показать текст на всех мониторах, только на главном или на любом отдельном мониторе.

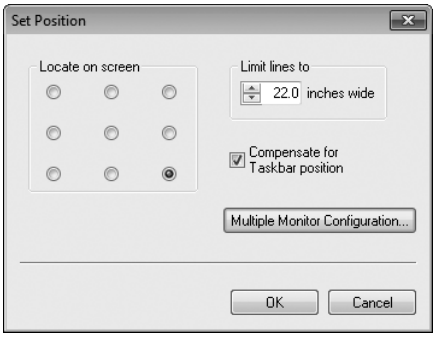


Рис. 10-6. Окно BgInfo Set Position

Можно настроить глубину цвета рисунка в меню **Bitmap**. Доступные варианты — 256 цветов (8 бит), 16 бит, 24 бит или **Match Display (По текущим настройкам)**.

Выберите **Location** в меню **Bitmap**, чтобы указать место записи созданных обоев. По умолчанию он создается в папке временных файлов пользователя. Обратите внимание на то, что для создания файла в папке Windows необходимы права администратора. При выборе **Other Directory (Другой**

каталог) можно включить в путь переменные окружения и имя конечного файла.

Для предварительного просмотра обоев, сгенерированных BgInfo, щелкните -переключатель **Preview (Предварительный просмотр)** — BgInfo покажет обои в полноэкранном режиме на основном дисплее. Если продолжить редактирование обоев, в режиме предварительного просмотра изменения будут видны сразу после выбора команды **Refresh** в меню **File** или нажатия F5.

Сохранение конфигурации BgInfo

Выберите **Save As** в меню **File**, чтобы сохранить в файл текущие настройки конфигурации BgInfo. Сохраненную конфигурации можно применить к другим рабочим столам пользователя или на других компьютерах, просто указав файл в командной строке BgInfo. Можно открыть файл конфигурации для дальнейшего редактирования, выбрав **File | Open**. Также можно открыть файл, дважды щелкнув его в **Проводнике**. При первом запуске BgInfo **Проводник** создает для BgInfo файловую ассоциацию для расширения .bgi.

При запуске BgInfo с файлом конфигурации в командной строке открывается редактор BgInfo с 10-секундным таймером задержки создания обоев. При добавлении в командную строку параметра **/timer:0** BgInfo создает обои без задержки, не открывая свое окно. Например, чтобы при входе в систему любого пользователя на рабочем столе отображалась обновленная информация о системе, можно создать в общей папке загрузки ярлык со следующей командной строкой:

```
Bginfo.exe c:\programdata\bginfo.bgi  /timer:0 /silent
```

Параметр **/silent** блокирует все сообщения об ошибках.

Помимо параметров отображения, файл конфигурации содержит определения пользовательских полей, информацию о рабочих столах и альтернативные параметры вывода данных (о них речь пойдет ниже). Выберите **Reset Default Settings (Восстановить настройки по умолчанию)** в меню **File**, чтобы удалить всю информацию о конфигурации и восстановить изначальное состояние BgInfo. Текущие настройки BgInfo хранятся в разделе реестра HKCU\Software\Winternals\BgInfo, за исключением параметра *EulaAccepted*, который хранится в разделе HKCU\Software\Sysinternals\BgInfo.

Другие параметра вывода данных

Поскольку BgInfo собирает много полезной информации, нам показалось вполне естественным добавить возможность ее сохранения не только в виде изображений. BgInfo может записывать собранные данные в файлы различных форматов или в базу данных Microsoft SQL Server. Также программа может отображать информацию в отдельном окне. Чтобы использовать эти параметры без изменения обоев, щелкните кнопку **Desktops** и выберите **Do Not Alter This Wallpaper (Не изменять обои)** для всех рабочих столов.

Чтобы сохранить данные в CSV-файл, таблицу Microsoft Excel или базу данных Access, выберите формат **Database (База данных)** в меню **File**. Когда откроется диалоговое окно **Database Settings (Настройки базы данных)**, изображенное на рис. 10-7, введите полный путь к файлу с расширением, соответственно, **.txt**, **.xls** или **.mdb**. Кнопка **File** открывает диалог сохранения. Для добавления записей в существующий файл выберите **Create A New Database Record For Every Run (Создавать новую запись в базе данных для каждого запуска)**. Чтобы хранить только одну запись для текущего компьютера, выберите **Record Only The Most Recent Run For Each Computer (Регистрировать только последний запуск для каждого компьютера)**. Можно сохранить эту конфигурацию в файл с расширением .bgi и применить ее позднее, либо просто щелкнуть **OK** или **Apply** в главном окне BgInfo. Обратите внимание на то, что вывод включает не только поля, выбранные для отображения, но и все поля по умолчанию, а также пользовательские поля.

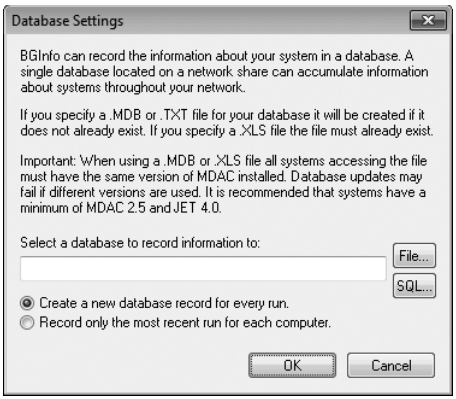


Рис. 10-7. Окно Database Settings

Для записи данных в базу данных SQL Server выберите **Database** в меню **File**, щелкните кнопку **SQL**, выберите экземпляр SQL Server, а затем выберите **Use Trusted Connection (Использовать доверенное соединение)** (для входа в Windows) или введите имя учетной записи и пароль (для унаследованной стандартной аутентификации SQL). Нужно выбрать имеющуюся базу данных в разделе **Options** диалогового окна **SQL Server Login (Вход в SQL Server)**, как показано на рис. 10-8. При первом вводе информации BgInfo создает и настраивает таблицу в базе данных с именем, указанным вами в поле **Application Name (Имя приложения)**. BgInfo создает столбец даты и времени с временной меткой и столбец nvarchar(255), соответствующий каждому полю по умолчанию и пользовательскому полю. Эти однократные операции требуют наличия у первого вызывающего пользователя разрешений CREATE TABLE и ALTER. После создания таблицы вызывающим пользователям потребуется разрешение CONNECT для базы данных, а также разрешения SELECT, INSERT и UPDATE для таблицы.

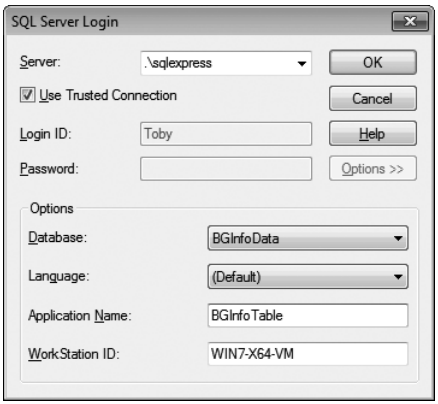


Рис. 10-8. Конфигурация BgInfo для записи в таблицу SQL Server

Для записи данных в файл формата .RTF запустите BgInfo с параметром командной строки **/rtf:**path и укажите файл конфигурации BgInfo (.bgi). Обратите внимание на то, что эта функция включает форматирование текста, но не фона. По этой причине вам придется изменить цвет текста, который по умолчанию является белым. Можно также добавить в командную строку параметр **/timer:0**, чтобы пропустить 10-секундный отсчет.

Чтобы отобразить данные BgInfo во всплывающем окне, а не на обоях, добавьте в командную строку BgInfo параметр **/popup**. Чтобы отобразить значок BgInfo в области уведомлений на панели задач, добавьте в командную строку параметр **/taskbar**. Чтобы открыть всплывающее окно BgInfo, щелкните этот значок.

Обновление других рабочих столов

В Windows XP и Windows Server 2003 BgInfo может изменять фоновый рисунок рабочего стола, который появляется перед входом пользователя в систему. Щелкните кнопку **Desktops**, чтобы открыть диалоговое окно **Desktops** (рис. 10-9). Можно отдельно задать изменение обоев рабочего стола текущего пользователя, рабочего стола для консольных пользователей и рабочего стола для пользователей служб терминалов (т.н. удаленные рабочие столы). Можно также выбрать замену обоев для всех рабочих столов **None (Никакие)**. Учтите, что изменение рабочих столов требует привилегий администратора, и данная функция не работает в Windows Vista и выше. Можно включить вывод программой BgInfo сообщения об ошибке, если проблемы с разрешениями мешают ей обновить рабочий стол входа.

Если на компьютере существует несколько интерактивных сеансов, включая сеансы удаленного рабочего стола и быстрого переключения между пользователями, можно обновить обои рабочих столов всех пользователей с помощью параметра командной строки **/all**.

Рис. 10-9. Диалоговое окно Desktops

При добавлении параметра **/all** BgInfo запускает службу, которая перечисляет текущие интерактивные сеансы и запускает экземпляр BgInfo в каждом сеансе под учетной записью пользователя, управляющего сеансом. В связи с тем, что экземпляры BgInfo запускаются в контексте разных пользователей, нужно указывать файл конфигурации по абсолютному пути и в папке, доступной для чтения всем пользователям. Кроме того, нужно добавить в командную строку параметры **/accepteula** и **/timer:0**.

Desktops

Утилита Desktops компании Sysinternals дает возможность управлять приложениями на четырех виртуальных рабочих столах. Можно читать электронную почту на одном рабочем столе, просматривать веб-страницы на втором и работать с различными программами на третьем, избегая при этом путаницы с окнами. После настройки клавиатурных комбинаций для переключения между столами можно создавать рабочие столы и переключаться между ними щелчком значка утилиты в области уведомлений (так открывается режим предварительного просмотра и окно переключения) либо с помощью клавиатурных комбинаций.

В отличие от других программ для управления виртуальными рабочими столами, которые отображают активные окна и скрывают остальные, Desktops использует объект рабочего стола Windows для всех рабочих столов. При создании окна приложений связываются с объектом рабочего стола, поэтому Windows «знает», какие окна с каким столом связаны и отображает «нужные» окна при переключении на тот или иной рабочий стол. Благодаря этому программа Desktops обладает малыми размерами и лишена багов, присутствующих в других программах, в которых при отображении активных окон и нарушается связь с другими окнами (см. главу 2).

При первом запуске Desktops открывается окно конфигурации (рис. 10-10). В нем настраиваются клавиатурные комбинации для переключения между рабочими столами и автоматический запуск Desktops во время

входа в систему. Чтобы вызвать окно конфигурации, щелкните правой кнопкой мыши области уведомлений Desktops и выберите **Options**.

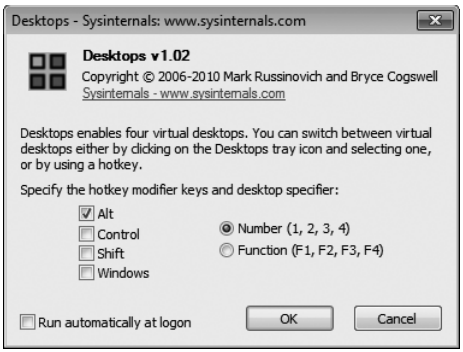


Рис. 10-10. Диалоговое окно конфигурации Desktops

Для переключения между рабочими столами щелкните значок Desktops в области уведомлений — откроется окно переключения (рис.10-11) со значками четырех доступных рабочих столов. При первом запуске Desktops создается только первый рабочий стол. Если щелкнуть другой рабочий стол, Desktops создаст его, запустит на нем Проводник и переключится на него. Быстрее можно переключаться с помощью клавиатурных комбинаций (например, Alt+3 для переключения на Рабочий стол 3). Переключившись на рабочий стол, можно запускать на нем приложения. Значок в области уведомлений Desktops отображает активный рабочий стол во всплывающей подсказке. Заметьте, что темы и обои, заданные для одного стола, применяются ко всем четырем рабочим столам.

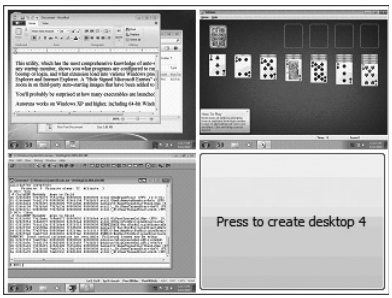


Рис. 10-11. Окно переключения Desktops

Поскольку в Desktops используются объекты рабочего стола Windows, утилита не поддерживает некоторые функции других аналогичных утилит. Например, Windows не позволяет перемещать окна между столами; кроме того, для управления панелью задач и меню Пуск на каждом рабочем столе нужен отдельный процесс Explorer, большинство значков области уведомлений отображаются только на первом рабочем столе. В настоящее время интерфейс Aero также работает только на первом рабочем столе. Кроме того, удалить объект рабочего стола невозможно, поэтому Desktops не позволя-

ет закрывать рабочие столы, так как это приведет к зависанию окон и процессов. Поэтому лучше выйти из Desktops, чтобы закрыть рабочие столы. Заметьте, что при выходе из системы на любом рабочем столе выход будет выполнен и на остальных столах.

Утилита Desktops от Sysinternals совместима со всеми поддерживаемыми версиями Windows и с сеансами удаленных рабочих столов.

ZoomIt

ZoomIt — это программа для увеличения размера и аннотирования экрана. Изначально я написал ее для себя: программа облегчает демонстрацию презентаций PowerPoint и приложений. Помимо работы с презентациями я использую ее для быстрого увеличения части экрана и захвата увеличенных экранных снимков с аннотациями.

ZoomIt работает в фоновом режиме и активируется пользовательской клавиатурной комбинацией для увеличения или уменьшения части экрана, отдельные комбинации служат для рисования и ввода текста на увеличенном изображении. Программа также выводит таймер обратного отсчета, который я запускаю во время перерывов длительных занятий, чтобы слушатели знали, когда продолжится занятие.

У ZoomIt два режима. В стандартном режиме нажатие клавиатурной комбинации создает снимок рабочего стола, а в режиме LiveZoom — увеличивает рабочий стол, в то время как программы продолжают работать и обновлять экран в режиме реального времени.

ZoomIt работает во всех поддерживаемых версиях Windows, а на планшетных компьютерах можно рисовать в ZoomIt с помощью пера.

Работа с ZoomIt

При первом запуске ZoomIt программа открывает окно конфигурации (рис. 10-12) с описанием всех функций и настройкой комбинаций для них. После подтверждения изменений щелчком **OK** или отмены (**Cancel**) ZoomIt продолжает работать в фоновом режиме. Чтобы снова открыть окно конфигурации, щелкните значок ZoomIt в области уведомлений и выберите **Options** в меню. Если вы отменили отображение значка, запустите ZoomIt снова.

По умолчанию комбинация Ctrl+1 служит для масштабирования, Ctrl+2 запускает режим рисования без масштабирования, Ctrl+3 включает таймер перерыва, а Ctrl+4 запускает LiveZoom. Дальнейшее изложение предполагает, что сохранены настройки по умолчанию. В системах с несколькими мониторами ZoomIt работает на «активном» дисплее, то есть, на том, где указатель мыши находился во время нажатия комбинации, остальные мониторы продолжают работать в обычном режиме.

Во всех режимах ZoomIt, за исключением LiveZoom, нажатие Ctrl+C копирует в буфер обмена содержимое текущего экрана, включая аннотации. Можно также сохранять содержимое дисплея в файл формата PNG (нажа-

тием Ctrl+S, при этом ZoomIt запрашивает путь и имя файла для сохранения изображения).

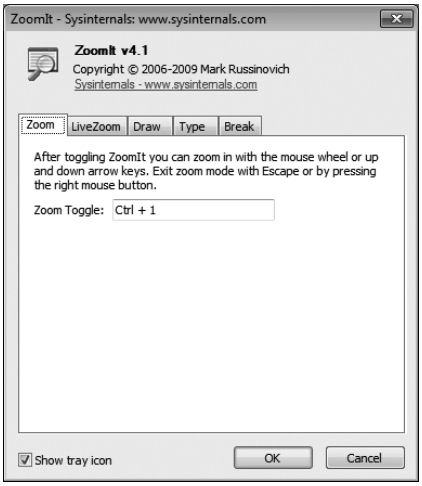


Рис. 10-12. Вкладка Zoom в окне конфигурации ZoomIt

Режим масштабирования

Для включения стандартного режима масштабирования нажмите Ctrl+1. ZoomIt захватывает снимок экрана и увеличивает его в два раза относительно текущего положения указателя мыши. Можно увеличивать или уменьшать снимок нажатием клавиш-стрелок вверх и вниз или прокручивая колесико мыши. Также можно перемещать точку, относительно которой масштабируется снимок, перетаскивая указатель мыши.

Из стандартного режима можно перейти в режим рисования, щелкнув левой кнопкой, или в режим печати, нажав клавишу T.

Для выхода из стандартного режима нажмите клавишу Escape или щелкните правой кнопкой.

Режим рисования

В режиме рисования можно рисовать на экране прямые и произвольные контуры и фигуры различных цветов и различной толщины (рис. 10-13). Можно также очистить экран, превратив его в белую или черную доску для рисования.

Для рисования произвольных линий формы переместите указатель мыши в точку начала линии, а затем, удерживая левую кнопку, перетащите мышь. Чтобы прекратить рисование, отпустите кнопку мыши — ZoomIt продолжит работу в режиме рисования. Для выхода из режима рисования щелкните правой кнопкой мыши.

Чтобы нарисовать прямую, установите указатель мыши в начало прямой. Нажмите и удерживайте клавишу Shift, а затем, удерживая левую кнопку

мыши, перетащите указатель в конечную точку линии — прямая будет отображаться на экране во время перемещения указателя, пока вы не отпустите кнопку мыши в конечной точке линии. Если линия располагается почти горизонтально (или вертикально), ZoomIt автоматически сделает ее горизонтальной (или вертикальной). Чтобы нарисовать стрелку, установите указатель в точку, где будет острие стрелки, затем, удерживая Shift+Ctrl и левую кнопку мыши, перетащите указатель в точку начала стрелки.

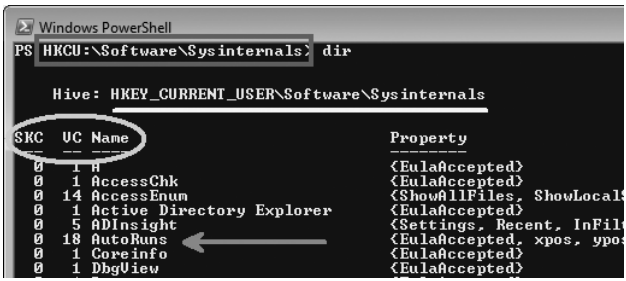


Рис. 10-13. Режим рисования в ZoomIt

Чтобы нарисовать прямоугольник, установите указатель мыши в левый верхний или правый нижний угол. Удерживая Ctrl и левую кнопку мыши, перетащите указатель. Размер прямоугольника будет изменяться в процессе перетаскивания, пока вы не отпустите кнопку мыши, после чего прямоугольник появится на экране. Чтобы нарисовать эллипс, выполните те же действия, но удерживайте клавишу Tab вместо Ctrl. Начальная и конечная точки перетаскивания прямоугольник, в который будет вписан эллипс.

Для отмены рисования последнего элемента нажмите Ctrl+Z. Для удаления всех аннотаций нажмите e.

Чтобы превратить экран в белую доску нажмите, в черную доску экран превращается при нажатии k. Для перехода в режим печати нажмите t.

В режиме рисования, можно изменять цвет пера: нажатие r устанавливает красный цвет, g — зеленый, b — синий, o — оранжевый, y — желтый, а p — розовый. Цвет пера действует и в режиме печати.

Ширина пера изменяется нажатием левой клавиши Ctrl и стрелки вверх или вниз либо колесиком мыши.

Режим печати

Для перехода из режима масштабирования или рисования в режим печати нажмите t. Указатель мыши примет форму вертикальной линии, указывающей размер, положение и цвет текста. Для изменения положения перетащите указатель мыши, а для изменения размера шрифта прокрутите колесико мыши или нажмите клавишу-стрелку вверх или вниз. Шрифт можно изменить на вкладке **Type** окна **Options**. Чтобы задать начальную точку ввода текста щелкните левой кнопкой мыши или просто начните печатать (рис. 10-14). Для выхода из режима печати нажмите Esc.

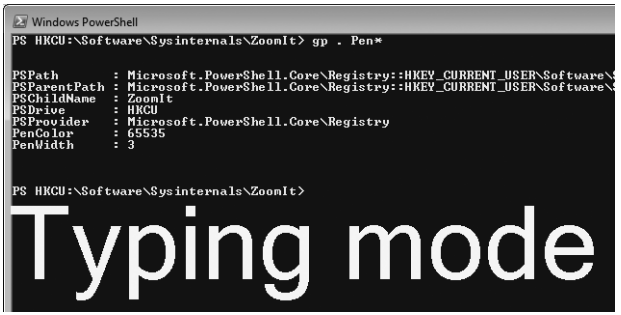


Рис. 10-14. Режим печати ZoomIt

Таймер

Таймер включается нажатием клавиш Ctrl+3. По умолчанию таймер отсчитывает 10 минут. Во время работы таймер регулируется нажатием клавиш-стрелок вверх и вниз (так регулируется число минут) и влево–вправо (эти нажатия увеличивают либо уменьшают время таймера шагами в 10 секунд). Запуск таймера настраивается на вкладке **Break (Перерыв)** в окне **Options**. Шрифт таймера тот же, что задан в режиме печати.

Параметр **Show Time Elapsed After Expiration (Считать время по окончании обратного отсчета)** определяет, остановится ли отсчет, дойдя до нуля, или продолжится с отрицательными значениями. Кнопка **Advanced** позволяет настраивать дополнительные параметры, в том числе звуковой сигнал таймера, его прозрачность и положение, а также отображение фонового рисунка или текущего рабочего стола за таймером вместо белого фона по умолчанию.

Режим LiveZoom

В обычном режиме масштабирования делается снимок текущего рабочего стола, который затем масштабируется и аннотируется, а в LiveZoom увеличивается реальный рабочий стол, в то время как программы продолжают работать и обновлять экран в режиме реального времени. Режим LiveZoom поддерживается в Windows Vista и выше. Наиболее эффективно он работает при включенном интерфейсе Aero.

В режиме LiveZoom ввод мышью и с клавиатуры поступает работающим программам, а не снимку, поэтому в нем не работают режимы рисования и печати. Кроме того, поскольку клавиши-стрелки и Esc используются в приложениях, в LiveZoom стрелки работают в комбинации с Ctrl (для масштабирования), для выхода из режима LiveZoom отдельная клавиша. Кроме того, участки экрана масштабируются мышью, только когда ее указатель находится рядом с краями экрана.

Из режима LiveZoom можно быстро переключаться в режим рисования нажатием Ctrl+1, вернуться в в LiveZoom позволяет нажатие Esc. Эта функция также наиболее эффективно работает при включенном интерфейсе Aero.

Глава 11

Утилиты для работы с файлами

В этой главе рассматриваются утилиты Sysinternals, предназначенные для управления файлами. Все эти утилиты являются консольными:

- **Strings** выполняет поиск встроенного текста ASCII и Unicode в файлах;
- **Streams** выявляет объекты файловой системы, имеющие альтернативные потоки данных, и, если указано, удаляет эти потоки;
- **Junction** и **FindLinks** показывают и позволяют управлять соединениями папок и жесткими ссылками (два типа ссылок NTFS);
- **DU** показывает логический размер иерархии папок и ее размер на диске;
- **PendMoves** и **MoveFile** показывают и регистрируют операции с файлами, которые будут выполнены при следующей загрузке системы.

Strings

В программировании термин «string» (строка) обозначает структуру данных, состоящую из последовательности символов, обычно представляющих текст, понятный человеку. Существуют различные утилиты, позволяющие искать в файлах встроенные строки. Однако многие из них (например, утилита *findstr*, входящая в состав Windows) ищут только текст в формате ASCII и игнорируют текст Unicode, тогда как другие (например, *find*, тоже входящая в состав Windows) некорректно выполняют поиск в двоичных файлах. Утилита Strings из пакета Sysinternals не имеет подобных проблем, что делает ее полезной для поиска файлов, а также для исследования содержимого неизвестных файлов в поисках строк, которые могут прояснить происхождение и назначение этих файлов.

Синтаксис командной строки Strings:

```
strings [-a] [-b байты] [-n длина] [-o] [-q] [-s] [-u] файл_или_папка
```

Параметр *файл_или_папка* обязателен. Он принимает знаки подстановки (например, *.dll). Strings выполняет поиск всех подходящих файлов и по умолчанию выводит в стандартный поток вывода все найденные строки ASCII и Unicode длиннее трех символов, в порядке их вхождения в теле файла. Чтобы искать только строки ASCII или Unicode, используйте пара-

метры -a или -u, соответственно. Параметр -s выполняет рекурсивный поиск папок. Чтобы указать минимальную длину строки (по умолчанию 3), укажите ее после параметра -n. Если добавить параметр -o, будет сообщаться смещение в файле, по которому начинается строка. Чтобы выполнять поиск только в начале файла, используйте параметр -b для ограничения числа проверяемых байтов. Наконец, параметр -q (quiet, тихий режим) убирает заголовок Strings из выходных данных; это полезно, когда выходные данные будут обрабатываться другой утилитой.

Следующая команда выполняет в первых 850000 байтах explorer.exe поиск строк Unicode длиной не менее 20 символов, убирая текст заголовка Strings. Полученные строки сортируются по алфавиту. На рис. 11-1 показана часть результатов.

```
strings -b 850000 -u -n 20 -q explorer.exe | sort
```

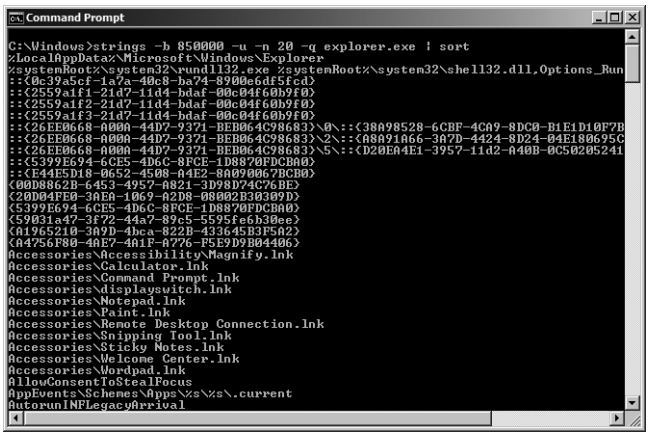


Рис. 11-1. Извлечение текста из explorer.exe при помощи Strings

Streams

Утилита Streams показывает объекты файловой системы, имеющие альтернативные потоки данных, и позволяет удалять их. NTFS предоставляет возможность создавать в файлах и папках альтернативные потоки данных. По умолчанию файл не имеет альтернативных потоков, и его содержимое хранится в основном неименованном потоке. Но если использовать синтаксис *имя_файла:имя_потока*, можно выполнять чтение и запись альтернативных потоков. Не все приложения имеют поддержку альтернативных потоков, но работать с такими потоками просто. Откройте командную строку, перейдите в папку на томе NTFS, в которую разрешена запись, и выполните следующую команду:

```
echo hello > test.txt:altdata
```

Вы только что создали поток с именем altdata, связанный с файлом test.txt. Обратите внимание: если просмотреть размер файла test.txt при помощи

команды DIR или в Проводнике, будет показан нулевой размер (если файл test.txt не существовал до выполнения указанной команды), а при открытии в текстовом редакторе этот файл будет пустым. (Начиная с Windows Vista, для просмотра сведений об альтернативных потоках можно использовать команду DIR /R.) Для просмотра содержимого альтернативного потока выполните следующую команду:

```
more < test.txt:altdata
```

Команды type и more не поддерживают синтаксис альтернативных потоков, но оболочка Cmd.exe и ее операции перенаправления поддерживают.

Windows применяет альтернативные потоки данных при обработке загруженных файлов. Служба выполнения прикреплений (Attachment Execution Service) Windows добавляет поток *Zone.Identifier*, в котором указывает зону безопасности, из которой получен файл, чтобы система могла продолжать обрабатывать этот файл в соответствии с правилами, установленными для этой зоны. Один из способов удаления этого индикатора из файла — открыть его диалоговое окно **Свойства (Properties)** в Проводнике и щелкнуть кнопку **Разблокировать (Unblock)**. Однако это кнопка и другие элементы пользовательского интерфейса, позволяющие удалить информацию о зоне безопасности, часто скрыты групповой политикой.

Утилита Streams проверяет указанные файлы и папки и сообщает имена и размеры всех обнаруженных альтернативных потоков. Можно выполнить поиск по структурам папок и получить список файлов и папок с альтернативными потоками. Можно также удалить эти потоки, например, чтобы разблокировать загруженное содержимое. Для этого используется следующий синтаксис:

```
streams [-s] [-d] файл_или_папка
```

Параметр файл_или_папка является обязательным и принимает знаки подстановки. Например, команда streams *.exe проверяет все системные объекты, имя которых оканчивается на «.exe», расположенные в текущей папке, и выводит список тех из них, которые имеют альтернативные потоки данных. Выходные данные выглядят следующим образом:

```
C:\Users\Abby\Downloads\msvbvm50.exe:
   :Zone.Identifier:$DATA       26
```

В этом примере файл msvbvm50.exe имеет альтернативный поток «Zone. ldentifier» размером 26 байт. Чтобы просмотреть содержимое этого потока, используйте команду more < msvbvm50.exe:Zone.ldentifier.

Параметр -s выполняет рекурсивную проверку папок, а параметр -d удаляет найденные альтернативные потоки. Например, следующая команда выполняет поиск файлов с альтернативными потоками в папке Downloads пользователя Abby и удаляет все обнаруженные альтернативные потоки:

```
streams -s -d C:\Users\Abby\Downloads
```

Streams сообщает имена удаленных альтернативных потоков.

На рис. 11-2 показаны выходные данные Streams после обнаружения и удаления альтернативного потока данных Zone.Identifier из файла SysinternalsSuite.zip. Если удалить поток Zone.Identifier перед извлечением утилит из этого архива позволяет запускать их без предупреждений безопасности и просматривать справочные файлы в формате Compiled HTML (.chm).

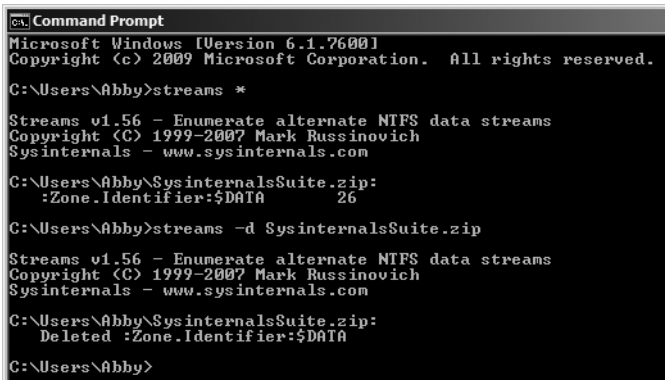


Рис. 11-2. Обнаружение и удаление альтернативных потоков данных

Утилиты для работы со ссылками NTFS

NTFS поддерживает как жесткие ссылки, так и мягкие, называемые также символьными ссылками. Жесткие ссылки доступны только для файлов, тогда как символьные ссылки можно использовать и с файлами, и с папками.

Жесткая ссылка позволяет создать несколько путей, ведущих на один файл на том же разделе. Например, если создать жесткую ссылку C:\Docs\Spec.docx, указывающую на существующий файл C:\Users\Abby\Documents\Specifications.docx, эти два пути будут вести к одному и тому же файлу на диске, который можно будет изменять, используя любой из этих путей. Для реализации жестких ссылок в NTFS ведется счет ссылок на файл, хранящийся на диске. При создании новой жесткой ссылки NTFS добавляет в данные ссылку с именем файла. Файл не удаляется, пока не обнулится его счетчик ссылок, поэтому можно удалить исходный файл (C:\Users\Abby\Documents\Specifications.docx в нашем примере) и продолжать использовать остальные жесткие ссылки (C:\Docs\Spec.docx). Файл, на который указывают жесткие ссылки, включает не только содержимое и альтернативные потоки данных, но и дескриптор безопасности, временные штампы и атрибуты, такие как «только чтение», «системный», «скрытый», «шифрованный» и «сжатый».

Символьные ссылки являются строками, которые интерпретируются динамически и могут быть относительными или абсолютными путями, ссылающимися на файл или папку на любом запоминающем устройстве, включая другие локальные тома и даже общие папки на других системах. Это значит,

что символьные ссылки не увеличивают счетчик ссылок исходного объекта файловой системы. При удалении исходного объекта данные удаляются, а символьные ссылки остаются, указывая на несуществующий объект. Символьные ссылки на файлы и папки имеют собственные разрешения и другие атрибуты, не зависящие от целевого объекта файловой системы.

Соединения очень похожи на символьные ссылки на папки, за тем исключением, что могут вести только к объектам на локальных томах. Соединения широко используются в Windows Vista и последующих версиях для обеспечения совместимости приложений. Например, в стандартной установке Windows 7 имя «C:\Documents and Settings» является соединением с C:\Users. Это позволяет работать многим программам, имеющим жестко прописанные пути. Разрешения на доступ к таким соединениям, обеспечивающим совместимость приложений, не позволяют просматривать содержимое соединения, так что программы резервного копирования, не поддерживающие соединения, не копируют одни и те же файлы несколько раз. Такие соединения обычно помечаются как скрытые и системные, поэтому по умолчанию не отображаются в списке папок.

В Windows Vista и последующих версиях ОС жесткие ссылки, символьные ссылки и соединения можно создавать при помощи команды mklink, встроенной в Cmd.exe. Пользователи, не имеющие административных прав, могут использовать mklink для создания жестких ссылок и соединений. Для создания символьных ссылок на файлы и папки требуется привилегия Create Symbolic Links (Создание символьных ссылок), по умолчанию предоставляемая только администраторам. В Windows XP и Windows Server 2003 команда mklink недоступна. Жесткие ссылки можно создавать также командой fsutil hardlink, а команда fsutil reparsepoint позволяет просматривать подробную информацию о соединениях и символьных ссылках и удалять их. Однако fsutil всегда требует административных прав.

В пакет Sysinternals включены две утилиты, заполняющие некоторые пробелы в управлении ссылками средствами Windows: Junction и FindLinks.

Junction

Junction позволяет создавать, удалять, искать соединения и отображать информацию о них. Если для работы с папкой, в которой создается или удаляется соединение, достаточно разрешений, Junction не требует административных прав и работает во всех поддерживаемых версиях Windows.

Синтаксис для создания соединения:

```
junction имя_соединения цель_соединения
```

имя_соединения — имя пути нового соединения, цель_соединения — существующая папка, на которую указывает соединение.

Синтаксис для удаления соединения:

```
junction -d имя_соединения
```

Удалить соединение можно также командой `rd`, встроенной в Cmd.exe. При использовании rd файлы и подпапки в целевой папке не удаляются, если не указать параметр /S.

Чтобы определить, является ли папка соединением, и если является, отобразить целевую папку, используйте следующую команду:

```
junction [-s] [-q] имя_соединения
```

имя_соединения указывает путь и может включать символы подстановки. Если имя указывает не на соединение, Junction возвратит сообщение «No reparse points found» («Точки повторного разбора не найдены»). Для выполнения поиска во вложенных папках используйте параметр -s. Параметр -q отключает сообщения об ошибках. Например, следующая команда показывает все точки соединения, найденные на диске C:

```
junction -s -q C:\
```

Следующая команда показывает все точки соединения в папке профиля пользователя:

```
junction %USERPROFILE%\*
```

Эта команда показывает все точки соединения из папки профиля пользователя, имена которых начинаются с «My»:

```
junction -s -q %USERPROFILE%\My*
```

На рис. 11-3 Junction показывает все точки соединения из папки ProgramData, обеспечивающие совместимость приложений.

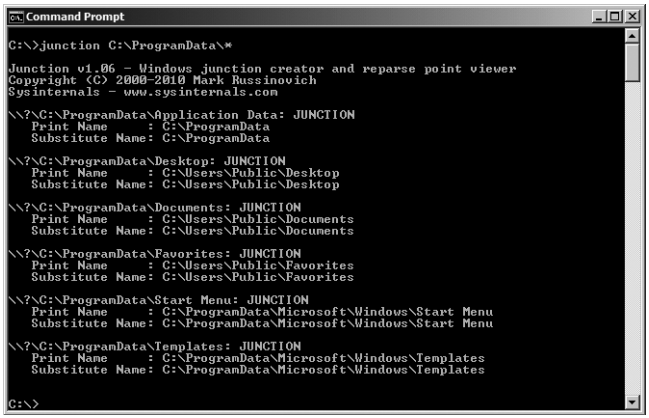


Рис. 11-3. Junction

FindLinks

FindLinks показывает жесткие ссылки, указывающие на данные файла. Просто выполните команду findlinks имя_файла; и если на указанный файл имеются другие жесткие ссылки, они будут показаны. Например, в Windows 7

x64 есть одна копия 64-разрядной версии Notepad.exe, на которую указывают жесткие ссылки из нескольких мест. На рис. 11-4 показаны выходные данные команд findlinks System32\Notepad.exe и findlinks SysWOW64\notepad.exe.

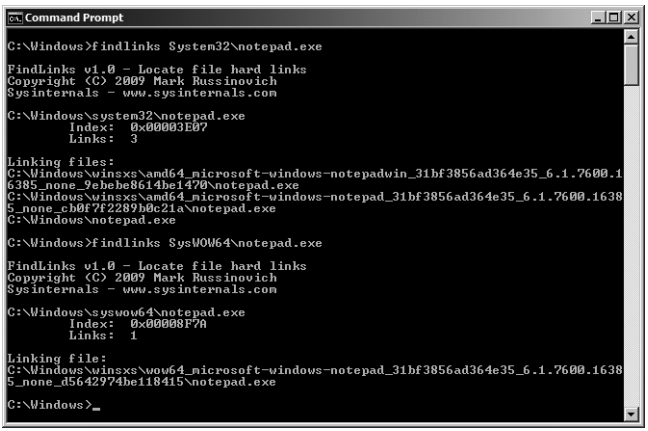


Рис. 11-4. FinlLinks

Видно, что четыре экземпляра Notepad.exe из папок Windows, System32 и двух папок winsxs в действительности являются одним файлом. Кроме того, есть 32-разрядная версия в папке SysWOW64, связанная с копией из папки winsxs. FindLinks показывает также индекс файла, 64-разрядный идентификатор, назначаемый файловой системой NTFS каждому уникальному файлу и папке на томе.

Начиная с Windows 7, можно искать жесткие ссылки, связанные с файлом, при помощи команды fsutil hardlink list имя_файла, но напомним, что fsutil всегда требует административных прав.

DU

Можно подумать, что подсчитать размер папки можно, просто просмотрев ее содержимое, пройдя через все подпапки и сложив размеры файлов. Однако все гораздо сложнее, так как для получения точного значения необходимо учесть жесткие ссылки, символьные ссылки на файлы и каталоги, соединения, сжатые и распределенные файлы, потоки альтернативных данных и неиспользуемое пространство кластера.

Утилита DU (Disk Usage, использование диска) показывает дисковое пространство, занятое иерархией папок, учитывая все эти факторы. По умолчанию она проходит подпапки, но не переходит в соединения и по символьным ссылкам на папки, а также игнорирует символьные ссылки на файлы. Она учитывает размер содержимого альтернативных потоков данных, включая потоки, связанные с объектами-папками. Файлы, на которые ссылается несколько жестких ссылок, считаются только один раз. Наконец, DU сообщает как логический размер, так и фактический размер на диске, учитывая

сжатые и распределенные файлы и неиспользуемое пространство кластера. Например, если папка содержит только один файл размером 10 байтов, DU покажет размер 10 байтов и «размер на диске» 4096 байтов, учитывая размер кластера, занятого файлом.

Синтаксис командной строки DU:

```
du [-n | -l уровни | -v] [-q] папка
```

По умолчанию DU проходит всю структуру целевой папки и отображает итоговые результаты, включая число обработанных файлов и папок, общий размер файлов и фактическое занятое место на диске. На рис. 11-5 показан пример выполнения команды du -q “C:\Program Files” (параметр -q убирает заголовок DU).

Рис. 11-5. Результаты команды du -q “C:\Program Files”

Параметры -n, -l и -v взаимно исключают друг друга. При использовании параметра -n DU не проходит подпапки и учитывает только файлы и папки, находящиеся только в самой целевой папке. Параметр -v используется для отображения размера в КБ промежуточных папок при их обработке. На рис. 11-6 показана часть результатов выполнения команды из предыдущего примера, но с параметром -v.

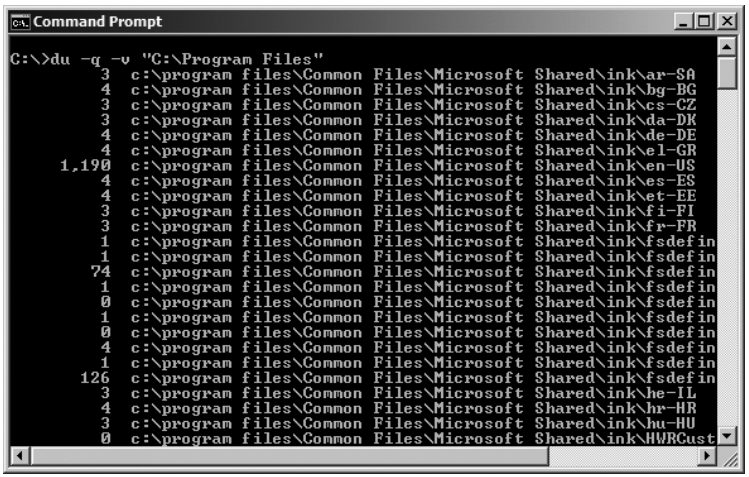


Рис. 11-6. Запуск DU с параметром -v

Параметр -l похож на -v. С ним сканируется вся иерархия папок, но промежуточные результаты сообщаются только для указанного числа уровней

папок. На рис. 11-7 показана часть результатов того же примера, но с использованием -l 1 вместо -v.

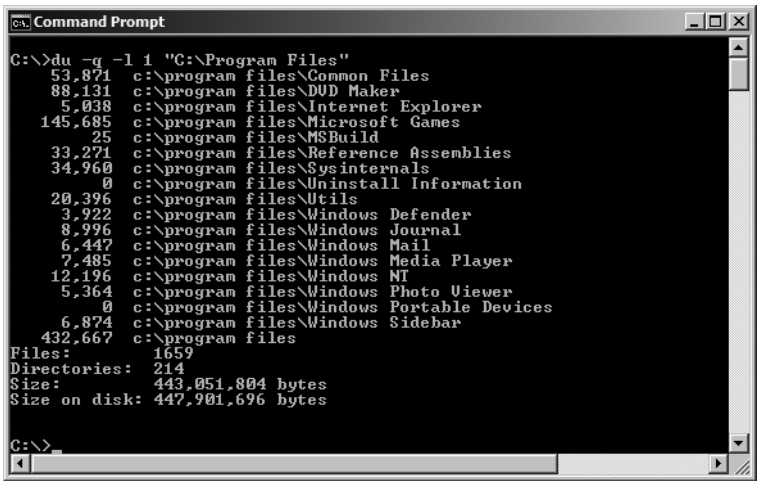


Рис. 11-7. Отображение промежуточных результатов для одного уровня папок

Утилиты для операций с файлами после загрузки

Программы-установщики часто не могут заменить, переместить или удалить файлы из-за того, что они используются. На этот случай Windows позволяет регистрировать такие операции, чтобы процесс диспетчера сеансов (Smss.exe) — первый процесс пользовательского режима, запускаемый в процессе загрузки, выполнял их при следующей загрузке системы, прежде чем какие-либо приложения или службы смогут занять файл и помешать его изменению. Приложения с администраторскими правами могут вызывать API *MoveFileEx* с флагом MOVEFILE_DELAY_UNTIL_REBOOT, который добавляет запросы на перемещение или удаление в значения *PendingFileRenameOperations* и *PendingFileRenameOperations2* типа REG_MULTI_SZ из раздела реестра HKLM\System\CurrentControlSet\Control\Session Manager.

PendMoves

PendMoves считывает значения *PendingFileRenameOperations* и *PendingFileRenameOperations2* и показывает все операции переименования и удаления файлов, которые будут выполнены при следующей загрузке. PendMoves также проверяет наличие исходного файла и отображает сообщение об ошибке, если он недоступен. Наконец, PendMoves показывает дату и время последнего изменения содержимого раздела Session Manager. Это позволяет понять, когда были зарегистрированы операции удаления и переименования.

Приведем пример выходных данных PendMoves, показывающих одну ожидающую операцию удаления файла и две операции перемещения, источник одной из которых не найден:

```
Source: C:\Config.Msi\3ec7bbbf.rbf
Target: DELETE

Source: C:\Windows\system32\spool\DRIVERS\x64\3\New\mxdwdrv.dll
Target: C:\Wi ndows\system32\spool\DRIVERS\x64\3\mxdwdrv.dll

Source: C:\Windows\system32\spool\DRIVERS\x64\3\New\XPSSVCS.DLL
*** Source file lookup error: The system cannot find the file specified.
Target: C:\Windows\system32\spool\DRIVERS\x64\3\XPSSVCS.DLL

Time of last update to pending moves key: 8/29/2010 11:55 PM
```

MoveFile

MoveFile позволяет запланировать операции перемещения, переименования и удаления файлов на следующую перезагрузку системы. Просто укажите имя существующей папки или файла и имя цели. Если объект нужно удалить, используйте в качестве имени цели две двойные кавычки. При помощи MoveFile можно удалить только пустую папку. Перемещение можно выполнять только в пределах одного тома, и при этом целевая папка должна существовать. Заметьте, что переименование — это перемещение, при котором не изменяется папка.

MoveFile требует административные права. В некоторых случаях отложенные операции с файлами могут не быть выполнены; об этом см. статью Knowledge Base 948601 (*http://support.microsoft.com/kb/948601*).

В следующем примере файл sample.txt перемещается из c:\original в c:\newdir после перезагрузки системы, если c:\newdir существует:

```
movefile c:\original\sample.txt c:\newdir\sample.txt
```

В этом примере файл sample.txt перемещается и переименовывается:

```
movefile c:\original\sample.txt c:\newdir\renamed.txt
```

В этом примере сначала удаляется файл c:\original\sample.txt, а затем папка c:\original, в которой не должно оставаться содержимого:

```
movefile c:\original\sample.txt «»
movefile c:\original «»
```

Глава 12

Утилиты для работы с диском

Предмет этой главы — утилиты для управления дисками и томами:

- **Disk2Vhd** позволяет получить образ физического диска в формате VHD;
- **Diskmon** выполняет мониторинг активности жесткого диска на уровне секторов в реальном времени;
- **Sync** сбрасывает модифицированные данные из дисковых кешей на диски;
- **DiskView** показывает графическую карту кластеров тома, позволяя выяснить, какой файл находится в указанных кластерах, и наоборот, какие кластеры занимает указанный файл;
- **Contig** позволяет просматривать степень фрагментации отдельных файлов и дефрагментировать их;
- **PageDefrag** дефрагментирует во время загрузки системные файлы, которые не могут быть дефрагментированы во время работы системы;
- **DiskExt** показывает информацию о экстентах диска;
- **LDMDump** показывает подробную информацию о динамических дисках, полученную из базы данных диспетчера логических дисков (Logical Disk Manager, LDM);
- **VolumeID** позволяет изменять идентификатор тома, называемый также *серийным номером* тома.

Disk2Vhd

Disk2Vhd создает образ физического диска в формате Virtual Hard Disk, VHD. Формат VHD поддерживается виртуальными машинами (VM) Hyper-V, Virtual PC и Virtual Server. Разница между Disk2Vhd и другими утилитами для преобразования физического диска в виртуальный заключается в том, что Disk2Vhd может создать образ системы Windows во время ее работы. Disk2Vhd использует функцию Volume Snapshot (моментальный снимок тома), появившуюся в Windows XP для создания моментальных снимков дисков, подлежащих преобразованию. С помощью Disk2Vhd возможно даже создавать VHD-образы диска, записывая образ на тот же самый

диск, но все же утилита работает быстрее, когда VHD-файл записывается на другой диск.

Disk2Vhd работает во всех поддерживаемых версиях Windows и требует администраторских прав.

Интерфейс Disk2Vhd отображает список томов, имеющихся в системе (рис. 12-1), и объем дискового пространства, необходимого для преобразования каждого из этих томов в VHD. Чтобы создать VHD-образ, просто выберите нужные тома, укажите путь к файлу, в который будет выполнена запись, и его имя, после чего щелкните кнопку **Create (Создать)**.

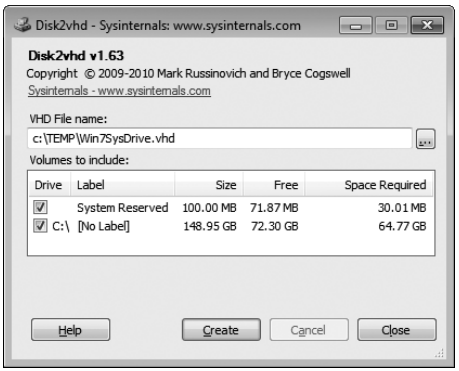


Рис. 12-1. Интерфейс Disk2Vhd

Будет создано по одному VHD-файлу на каждый диск, на котором находятся выбранные тома. В образ включается информация о разделах для всего диска, но копируются только с выбранных томов. Так удается, например, захватывать системные тома, исключая тома с данными. Для экономии места на диске и времени Disk2Vhd не копирует файлы подкачки и файлы спящего режима.

Чтобы использовать файлы VHD, полученные с помощью Dsik2Vhd, создайте виртуальную машину с нужными характеристиками и добавьте эти VHD в ее конфигурацию как диски IDE. При первой загрузке виртуальная машина с подключенной копией Windows обнаружит оборудование виртуальной машины и автоматически установит драйверы, если они имеются в образе. Если необходимых драйверов там нет, установите их через компоненты интеграции Virtual PC или Hyper-V. Можно также подключиться к VHD при помощи утилит управления дисками (Disk Management) или Diskpart из Windows Server 2008 R2 или Windows 7.

Если VHD-образ создается для Windows XP или Windows Server 2003 и будет использоваться как загрузочный VHD-диск в Virtual PC, установите флажок **Prepare For Use In Virtual PC (Подготовить для использования в Virtual PC)** (рис. 12-2), чтобы обеспечить совместимость уровня абстрагирования от аппаратных средств (Hardware Abstraction Layer), установленного в VHD, с Virtual PC. Этот флажок появляется только при запуске Disk2Vhd в Windows XP или Windows 2003.

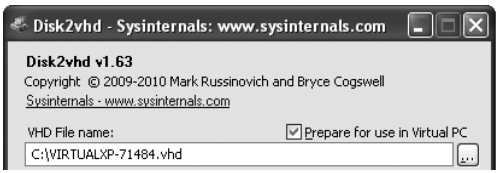


Рис. 12-2. Флажок Prepare For Use In Virtual PC при запуске Disk2Vhd в Windows XP

Параметры командной строки Disk2Vhd позволяют создавать VHD-образы при помощи сценариев:

```
disk2vhd [-h] диски файл_vhd
```

- -h — при создании образов системных томов Windows XP и Windows Server 2003 этот параметр исправляет HAL в VHD для совместимости с Virtual PC;
- диски — одна или несколько букв дисков с двоеточием (например, c: d:), указывают тома для создания образа. Можно использовать знак *, чтобы указать все тома;
- файл_vhd — полный путь к создаваемому VHD-файлу.

Приведем пример:

```
disk2vhd c: e:\vhd\snapshot.vhd
```

Максимальный поддерживаемый Virtual PC размер виртуального диска — 127 Гб. Если создать VHD на основе диска большего размера, он не будет доступен в виртуальной машине Virtual PC, даже если включить только данные с тома этого диска, имеющего меньший размер.

Заметьте также, что не следует подключать VHD-образ к тому же экземпляру Windows, в котором он был создан, если планируете загружаться с этого VHD: Windows назначает каждому подключенному диску уникальную сигнатуру. Если подключить VHD к системе с исходным диском, виртуальному диску во избежание конфликта будет назначена другая сигнатура. Windows ссылается на диски в базе данных конфигурации загрузки (boot configuration database, BCD) по сигнатурам, поэтому когда VHD получит новую сигнатуру, экземпляры Windows, загружаемые на виртуальной машине, не смогут найти виртуальный диск, указанный в BCD.

Примечание Средства переноса P2V Migration for Software Assurance преобразуют в виртуальный жесткий диск существующую клиентскую среду Windows XP и ОС следующих версий, используя Microsoft Deployment Toolkit, утилиту Disk2VHD и, при необходимости, System Center Configuration Manager 2007. Затем они автоматически устанавливают систему Windows 7, содержащую виртуальную машину с предыдущей средой Windows, приложениями и веб-обозревателем. Предыдущий виртуальный рабочий стол пользователя сохраняет имеющиеся компоненты управления, политики и членство в доменах. Кроме того, в процессе переноса приложения и обозреватель публикуются в меню **Пуск (Start)** Windows 7 для быстрого доступа.

Diskmon

Diskmon — приложение с графическим интерфейсом, записывающее в журнал и показывающее в реальном времени всю активность жестких дисков систем Windows на уровне секторов. В отличие от Process Monitor, перехватывающего логические события, имеющие отношения к файлам (включая запросы, удовлетворяемые из кэша или кэшируемые в памяти для дальнейшей записи на диск), Diskmon сообщает о событиях, имеющих отношение к физическому диску. Для каждой операции чтения и записи на диск Diskmon показывает диск и номера секторов, объем прочитанных или записанных данных, а также длительность операции. Можно запустить Diskmon в виде значка в области уведомлений, который будет изменять цвет на зеленый, когда выполняется чтение, или красный, когда производится запись.

Diskmon требует администраторских прав и работает во всех поддерживаемых версиях Windows.

Как видно на рис. 12-3, Diskmon назначает каждому событию порядковый номер и отображает его в первом столбце. Если внутренние буферы Diskmon переполняются во время активной работы, в нумерации событий появляются пропуски.

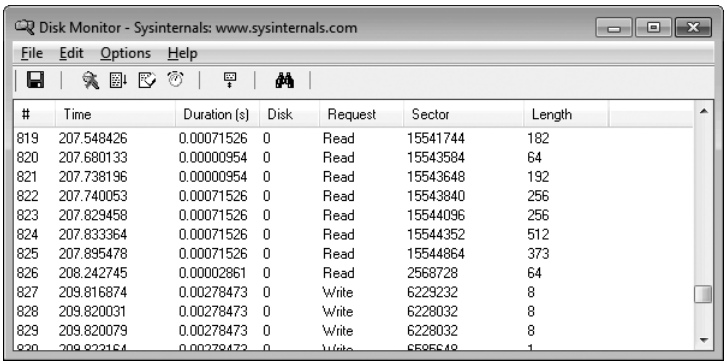


Рис. 12-3. Интерфейс Diskmon

В столбце **Time (Время)** показывается число секунд, прошедшее между началом отслеживания и инициацией запроса. Время начала сбрасывается, когда экран очищается нажатием Ctrl+X или щелчком кнопку **Clear (Очистить)** на панели инструментов. Вместо этого в меню **Options (Параметры)** можно выбрать отображение системного времени (с миллисекундами или без). В столбце **Duration (Длительность)** показано, как долго выполнялось чтение или запись, в секундах.

В столбце **Disk (Диск)** показываются начинающиеся с нуля номера *дисков*, а не разделов или томов, и с жестким диском может быть связано несколько томов. В столбце **Length (Длина)** показывается число записанных или прочитанных секторов. Большинство секторов жесткого диска имеют размер 512 байт.

Команды меню и кнопки панели инструментов работают так же, как и в других утилитах мониторинга Sysinternals. Чтобы включить или отключить захват данных, нажмите Ctrl+E. Чтобы записать показанные результаты в текстовый файл с разделением знаками табуляции, выберите в меню **File (Файл)** команду **Save (Сохранить)** или **Save As (Сохранить как)**. Чтобы на экране отображались последние захваченные события, включите автопрокрутку (**Autoscroll**). Поиск текста можно выполнить, нажав Ctrl+F, и повторить поиск нажатием F3. Для ограничения числа отображаемых событий задайте период журнала (**History Depth**). Можно выделить одну или несколько строк и скопировать их содержимое в буфер обмена, нажав Ctrl+C, или удалить, нажав Del. Также при помощи меню **Options (Параметры)** можно изменить шрифт и включить отображение окна Diskmon поверх остальных окон.

Чтобы запустить Diskmon в виде значка в области уведомлений на панели задач, выберите в меню **Options (Параметры)** команду **Minimize To Tray Disk Light (Свернуть в область уведомлений)** или укажите в командной строке параметр /l (строчная L). Если диск неактивен, значок будет серым; зеленым во время чтения с диска и красным во время записи. В этом режиме Diskmon не захватывает подробные сведения о событии для отображения в основном окне. Чтобы открыть окно Diskmon и начать захват данных, дважды щелкните значок в области уведомлений.

Sync

В большинстве Unix-систем имеется утилита *sync*, которая приказывает операционной системе сбросить на диск все измененные данные из буферов файловой системы. Это гарантирует, что данные из кэша файловой системы не будут потеряны при сбое в системе. Я написал эквивалент, тоже назвав его *Sync*, работающий во всех версиях Windows. Sync требует разрешения Write на целевом устройстве тома. Разрешения на запись в большинстве случаев даются только администраторам. Подробнее см. ниже во врезке «Разрешения для работы с томами».

Примечание После записи на съемный диск с файловой системой NTFS следует отключить том перед извлечением диска. По возможности, лучше использовать апплет Безопасное извлечение (Safe Removal), прежде чем извлекать какие-либо внешние запоминающие устройства.

Синтаксис командной строки Sync:

```
sync [-r | -e | буквы_дисков]
```

При запуске без параметров командной строки Sync перечисляет несъемные диски и очищает их. Если указать параметр -r или -e, будут также перечислены и очищены съемные диски; при использовании параметра -e съемные диски отключаются. Чтобы очистить конкретные диски, укажите их буквы. Например, чтобы очистить диски C и E, выполните команду `sync c e` (рис. 12-4).

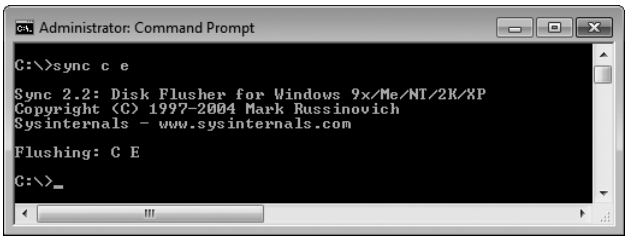


Рис. 12-4. Очистка дисков C и E

Разрешения для работы с томами Некоторые из описываемых в этой главе утилит требуют разрешений на целевом томе. Например, для выполнения команды sync e требуется разрешение Запись (Write) на диске E. Разрешения томов отличаются от разрешений корневых каталогов этих томов и могут налагать ограничения даже на работу с томами с файловыми системами, не поддерживающими контроль доступа, такими как FAT.

В Windows XP и всех версиях Windows Server разрешения на запись на все тома предоставляются только администраторам. Начиная с Windows Vista, интерактивно вошедшие пользователи получают разрешение на запись на тома съемных устройств, таких как флэш-диски.

Windows не предоставляет средств для просмотра разрешений объектов томов. Для этого можно использовать утилиту AccessChk, выполняя команду accesschk \\.\x:, где x — буква диска того тома, который нужно проверить. AccessChk описывается в гл. 8.

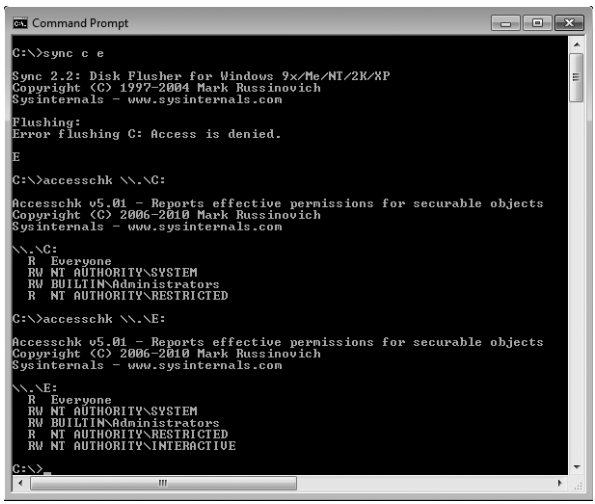


Рис. 12-5. Действие разрешений томов

На рис. 12-5 показана попытка очистить кеш дисков C и E при помощи утилиты Sync, запущенной в Windows 7 от имени обычного пользователя. Эта операция, требующая разрешения на запись, не выполнена для диска C, но выполнена для диска E. Затем запущена утилита AccessChk, показывающая действующие разрешения для этих двух томов. На диске C обычные пользователи имеют только разрешение на чтение, предоставленное группе **Everyone (Все)**, а на диске E интерактивные пользователи (NT AUTHORITY\INTERACTIVE) имеют разрешения Read (Чтение) и Write (Запись).

DiskView

DiskView показывает графическую карту кластеров тома NTFS, позволяя выяснить, в каких кластерах находится файл и фрагментирован ли он, или определить, каким файлом занят определенный сектор. DiskView требует администраторских привилегий и работает во всех поддерживаемых версиях Windows.

Запустите DiskView, выберите том из раскрывающегося списка **Volume (Том)** в левом нижнем углу окна и щелкните кнопку **Refresh (Обновить)**. DiskView просканирует весь том, заполняя цветные области окна (рис. 12-6). В нижней области представлен том целиком, слева кластер 0. Чтобы посмотреть значения цветовых кодов, выберите команду меню **Help | Legend (Справка | Легенда)**. Синим цветом показаны смежные кластеры файлов, красным — кластеры фрагментированных файлов, зеленым — кластеры системных файлов, белым — незанятые кластеры.

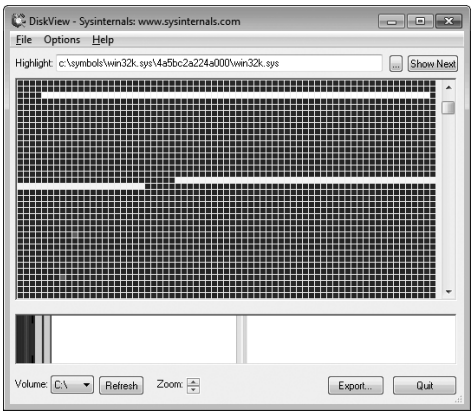


Рис. 12-6. DiskView

В верхней области представлена часть тома, которую можно выбрать щелчком в нижней области или вертикальной прокруткой. Часть, показанная в верхней области, выделена черными скобками в нижней области. Рекомендую развернуть окно, чтобы увидеть максимальную часть тома.

Примечание После сканирования тома DiskView может показать диалоговое окно **File Errors (Ошибки файлов)** со списком объектов, к которым не удалось обратиться. На рис. 12-7 показан типичный пример, когда не удалось получить доступ к файлу подкачки, потому что он использовался, и к папке System Volume Information из-за установленных разрешений.

Каждая ячейка в верхней области представляет кластер тома. (По умолчанию размер кластера для NTFS-томов емкостью более 2 ГБ равен 4096 байт.) Кнопка со стрелкой вверх рядом с надписью **Zoom (Масштаб)** позволяет увеличить размер ячеек, чтобы было проще различать отдельные кластеры и выделять ячейки. Если прокрутить верхнюю область до конца вверх, верхняя строка будет представлять первые кластеры диска; нулевой

кластер будет представлен левой верхней ячейкой, а кластер 1 будет располагаться справа от него. Вторая строка представляет следующий набор кластеров и т. д.

Рис. 12-7. Диалоговое окно File Errors

Цветовые коды, принятые по умолчанию, показывают в верхней области распределение файлов на диске. Темно-синие ячейки обозначают первый набор кластеров, связанный с файлом, а остальные светло-синие ячейки представляют кластеры этого файла, смежные с первым набором. Красные ячейки указывают на начало второго и последующих фрагментов файла, а стоящие рядом синие ячейки представляют остальные кластеры этого фрагмента.

Если отключить параметр **Show Fragment Boundaries (Показывать границы фрагментов)** в меню **Options (Параметры)**, цвет первых ячеек не будет отличаться, и фрагменты целиком будут выделены красным. Хотя исторически дефрагментаторы отображали фрагментацию именно так, это дает слишком пессимистическое представление о ситуации с фрагментацией. Алгоритм дефрагментации в Windows 7 не пытается объединить фрагменты размером более 64 МБ, поскольку преимущества в этом случае незначительны, а затраты на перемещение данных растут.

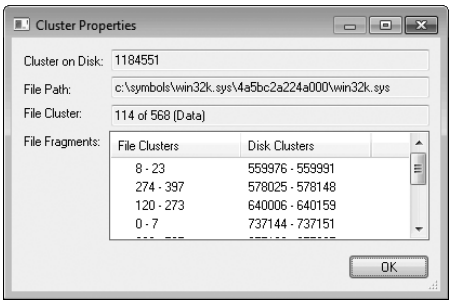


Рис. 12-8. Окно свойств кластера

Если щелкнуть цветную ячейке в верхней области, DiskView покажет имя файла, занимающего этот кластер, в текстовом поле вверху окна и выделит желтым цветом все кластеры, принадлежащие тому же файлу. Двойной щелчок ячейки открывает окно свойств кластера (**Cluster Properties**). Кроме

номера выделенного кластера и имени файла, занимающего этот кластер, в этом окне показан список фрагментов файла с номерами смежных кластеров, назначенных относительно файла (первый кластер файла имеет номер 0). В примере на рис. 12-8 файл занимает 568 кластеров, а выделенный кластер имеет номер 114.

Чтобы найти кластеры определенного файла, щелкните кнопку с точками, расположенной справа от текстовой области, и выберите файл. Первый фрагмент, принадлежащий этому файлу, будет выделен и станет видимым в верхней области. Щелкните кнопку **Show Next (Показать следующий)**, чтобы выделить и перейти к просмотру следующего фрагмента. Заметьте, что очень маленькие файлы могут храниться непосредственно в главной таблице файлов (Master File Table, MFT), а так как DiskView не анализирует файлы из MFT, если выбрать один из таких файлов, появится сообщение «The specified file does not occupy any cluster» («Выбранный файл не занимает кластеров»).

Команда **Statistics (Статистика)** из меню **File (Файл)** открывает диалоговое окно **Volume Properties (Свойства тома)**, в котором показано общее число файлов на томе, включая файлы в MFT, но в поле **Fragments (Фрагменты)** показано число фрагментов файлов, принадлежащих файлом, хранящимся вне MFT (рис. 12-9).

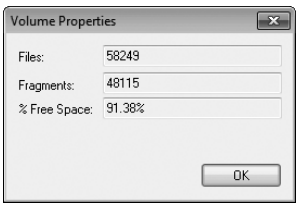


Рис. 12-9. Диалоговое окно свойств тома

Кнопка **Export (Экспорт)** записывает данные, полученные при сканировании, в текстовый файл, который можно импортировать в базу данных для дальнейшего анализа. Этот файл может получиться очень большим, так как содержит отдельную строку текста для каждого файла и каждого кластера на диске. Формат данных следующий:

- одна строка с числом файлов на диске;
- для каждого файла — одна строка с пробелом в качестве разделителя, содержащая следующее:
- число кластеров, занятых файлов;
- число фрагментов файла;
- путь к файлу;
- одна строка, содержащая число кластеров на диске;
- для каждого кластера — одна строка с пробелом в качестве разделителя, содержащая следующее:
- индекс файла (из предыдущего списка), занимающего кластер;

- индекс кластера в файле;
- тип кластера: 0 = данные, 1 = каталог, 2 = метаданные, 3 = неиспользуемый.

Contig

Большинство решений для дефрагментации дисков дефрагментирует том целиком. Contig — консольная утилита, позволяющая дефрагментировать отдельный файл или несколько файлов и просматривать уровень фрагментации файла. Возможность дефрагментировать определенный файл может быть полезной, когда этот файл постоянно фрагментируется из-за частой модификации. С помощью Contig можно также создать новый файл, который гарантированно будет располагаться в одном наборе смежных кластеров.

Contig работает во всех версиях Windows. Эта утилита использует стандартные API дефрагментации Windows, так что не приводит к повреждению диска даже при аварийной остановке. Contig требует разрешения Запись (Write) на целевом томе для выполнения дефрагментации файла; для просмотра уровня фрагментации требуется только разрешение Чтение (Read). Разрешения на запись требуются также для создания нового непрерывного файла. Подробнее см. выше во врезке «Разрешения для работы с томами».

Примечание Для твердотельных накопителей дефрагментация не требуется и только сокращает их время жизни.

Для работы с существующими файлами используйте следующий синтаксис:

```
contig [-v] [-q] [-s] [-a] имя_файла
```

Параметр имя_файла принимает знаки подстановки. Если параметр -a не используется, а целевой файл находится не в одном непрерывном блоке, Contig ищет на диске свободный блок, достаточно большой, чтобы вместить файл целиком, и, если находит, перемещает в него фрагменты файла. Файлы, уже находящиеся в непрерывном блоке, пропускаются. В конце дефрагментации Contig сообщает число обработанных файлов и число фрагментов файлов до и после операции.

С параметром -a Contig выполняет анализ файла или файлов, сообщает число фрагментов, но не перемещает их. Если указать параметр -v (verbose, подробный режим), будет показана дополнительная информация. С параметром -q (quiet, тихий режим) файлы обрабатываются в тихом режиме, в конце отображается только сводка. Если указать параметр -s и ввести имя файла со знаками подстановки, будет выполнен рекурсивный поиск в подкаталогах. Например, следующая команда анализирует фрагментацию всех файлов .bin в иерархии папок ProgramData:

```
contig -a -s C:\ProgramData\*.bin
```

Чтобы создать новый файл фиксированного размера, гарантированно расположенный в одном непрерывном блоке, используйте следующую команду:

```
contig -n имя_файла длина
```

На рис. 12-10 показан пример создания непрерывного файла размером 1 ГБ.

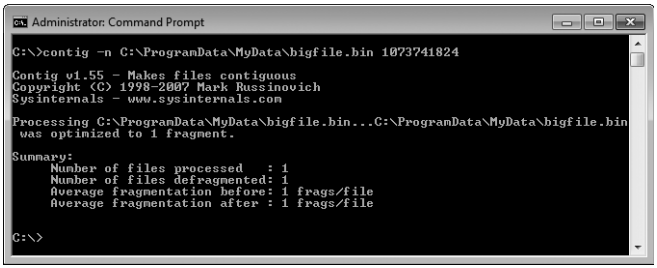


Рис. 12-10. Создание непрерывного файла при помощи Contig

PageDefrag

Одно из ограничений интерфейса дефрагментации Windows — невозможность дефрагментировать файл, открытый для эксклюзивного доступа. Таким образом, ульи реестра, файлы подкачки и файлы спящего режима нельзя дефрагментировать, пока Windows работает. Написанная мной утилита PageDefrag (PageDfrg.exe) обходит это ограничение. Она показывает текущий уровень фрагментации таких файлов и позволяет запланировать их дефрагментацию на начало цикла загрузки Windows.

Примечание PageDefrag работает только в версиях x86 Windows XP и Server 2003 и требует административных прав.

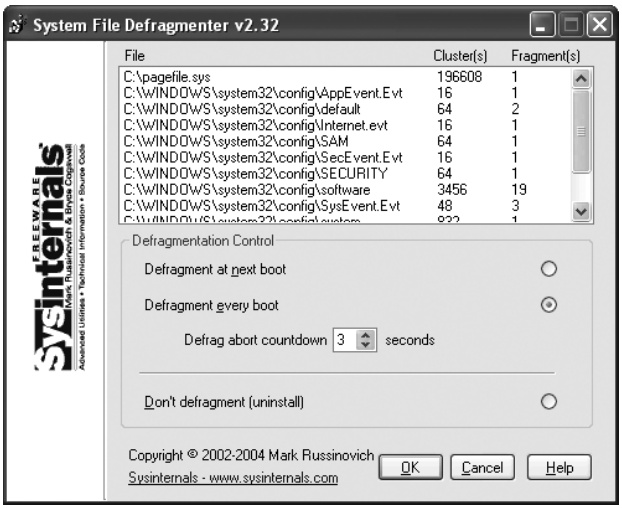


Рис. 12-11. Настройка PageDefrag

При запуске PageDefrag перечисляет системные файлы, которые не могут быть дефрагментированы, пока Windows работает, и показывает, сколько кластеров они занимают и на сколько фрагментов разделены (рис. 12-11). Установив соответствующий переключатель и щелкнув **OK**, можно указать, что эти файлы должны дефрагментироваться при следующей загрузке системы или при каждой загрузке. Можно также указать период ожидания перед запуском дефрагментации, в течение которого можно нажать любую клавишу, чтобы пропустить дефрагментацию.

Если выбрать дефрагментацию, PageDefrag зарегистрирует собственный исполняемый файл для запуска процессом диспетчера сеансов Windows в качестве программы BootExecute сразу после проверки Autochk. Данные о состоянии дефрагментации, выводимые в окне загрузки Windows, показаны на рис. 12-12.

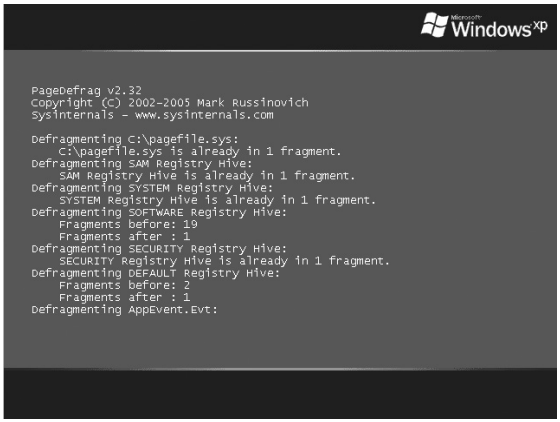


Рис. 12-12. Работа PageDefrag

PageDefrag можно выполнять из сценариев, используя следующий синтаксис:

```
pagedfrg { -e | -o | -n }
```

Параметр -e запускает дефрагментацию при каждой загрузке, -o — один раз при следующей загрузке, -n (never, никогда) отменяет запланированную дефрагментацию.

DiskExt

DiskExt — консольная утилита, показывающая информацию о том, на каких дисках и в каких физических местах разделов диска находятся разделы томов. (Тома могут распределяться по нескольким дискам.) При запуске без параметров DiskExt выводит список всех томов и информацию о них. Если указать в командной строке один или несколько томов, DiskExt сообщит только о них:

```
diskext c e
```

На рис. 12-13 показан пример запуска DiskExt без параметров.

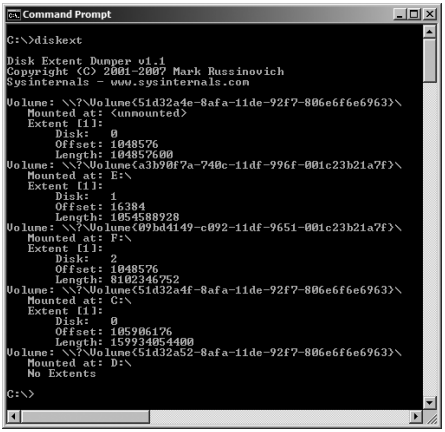


Рис. 12-13. DiskExt

Согласно определению из MSDN, «экстент диска — это непрерывный диапазон логических блоков, предоставленных диском. Экстент может представлять том целиком, одну часть составного тома, один компонент тома с чередованием или одну копию зеркалируемого тома». Каждый экстент начинается по смещению от начала диска, измеряемому в байтах, и имеет длину, также измеряемую в байтах.

DiskExt работает во всех поддерживаемых версиях Windows и не требует администраторских прав.

LDMDump

LDMDump — консольная утилита, отображающая подробную информацию о содержимом базы данных диспетчера логических дисков (Logical Disk Manager, LDM). В Windows реализована концепция *базовых* и *динамических* дисков. Динамические диски применяют более гибкую схему разделов по сравнению с базовыми. Динамическая схема поддерживает создание томов с несколькими разделами, предоставляющими функции обеспечения производительности, масштабируемости и надежности, не поддерживаемые простыми томами. К томам с несколькими разделами относятся зеркалируемые тома, массивы с чередованием (RAID-0) и массивы RAID-5. Разделы динамических дисков создаются при помощи LDM. LDM ведет единую унифицированную базу данных, в которой хранится информация о разделах всех динамических дисков системы. Эта БД располагается в зарезервированном пространстве размером 1 МБ в конце каждого динамического диска.

Примечание Подробнее об управлении томами и базе данных LDM см. в пятой редакции книги «Windows Internals», посвященной Windows Server 2008 и Windows Vista (Microsoft Press, 2009).

LDMDump принимает параметр /d# с номером диска, начинающимся с нуля:

```
Ldmdump /d0
```

Заметьте, что между /d и номером диска нет пробела.

Ниже показан фрагмент вывода LDMDump. Сначала показан заголовок базы данных LDM, затем идут записи из этой БД, описывающие том размером 12 ГБ с тремя динамическими дисками размером 4 ГБ. Запись базы данных тома указана как Volume1 (E:). В конце LDMDump перечисляет разделы и определения томов, хранящиеся в базе данных.

```
PRIVATE HEAD:
Signature           : PRIVHEAD
Version             : 2.12
Disk Id             : b5f4a801-758d-11dd-b7f0-000c297f0108
Host Id             : 1b77da20-c717-11d0-a5be-00a0c91db73c
Disk Group Id       : b5f4a7fd-758d-11dd-b7f0-000c297f0108
Disk Group Name     : WIN-SL5V78KD01W-Dg0
Logical disk start  : 3F
Logical disk size   : 7FF7C1 (4094 MB)
Configuration start : 7FF800
Configuration size  : 800 (1 MB)
Number of TOCs      : 2
TOC size            : 7FD (1022 KB)
Number of Configs   : 1
Config size         : 5C9 (740 KB)
Number of Logs      : 1
Log size            : E0 (112 KB)

TOC 1:
Signature           :   TOCBLOCK
Sequence            :   0x1
Config  bitmap start: 0x11
Config bitmap size  : 0x5C9
Log  bitmap start   : 0x5DA
Log bitmap size     : 0xE0
...
VBLK DATABASE:
0x000004:  [000001]  <DiskGroup>
         Name          : WIN-SL5V78KD01W-Dg0
         Object Id     : 0x0001
         GUID          : b5f4a7fd-758d-11dd-b7f0-000c297f010

0x000006:   [000003]  <Disk>
         Name          : Disk1
         Object Id     : 0x0002
         Disk Id       : b5f4a7fe-758d-11dd-b7f0-000c297f010
```

```
0x000007 :  [000005]  <Disk>
         Name          : Disk2
         Object Id     : 0x0003
         Disk Id       : b5f4a801-758d-11dd-b7f0-000c297f010

0x000008:   [000007]  <Disk>
         Name          : Disk3
         Object Id     : 0x0004
         Disk Id       : b5f4a804-758d-11dd-b7f0-000c297f010

0x000009: [000009] <Component>
         Name          : Volume1-01
         Object Id     : 0x0006
         Parent Id     : 0x0005

0x00000A: [00000A] <Partition>
         Name          : Disk1-01
         Object Id     : 0x0007
         Parent Id     : 0x3157
         Disk Id       : 0x0000
         Start         : 0x7C100
         Size          : 0x0 (0 MB)
         Volume Off    : 0x3 (0 MB)

0x00000B:   [00000B]  <Partition>
         Name          : Disk2-01
         Object Id     : 0x0008
         Parent Id     : 0x3157
         Disk Id       : 0x0000
         Start         : 0x7C100
         Size          : 0x0  (0 MB)
         Volume Off    : 0x7FE80003   (1047808 MB)

0x00000C:   [00000C]  <Partition>
         Name          : Disk3-01
         Object Id     : 0x0009
         Parent Id     : 0x3157
         Disk Id       : 0x0000
         Start         : 0x7C100
         Size          : 0x0  (0 MB)
         Volume Off    : 0xFFD00003   (2095616 MB)

0x00000D: [00000F] <Volume>
         Name          : Volume1
         Object Id     : 0x0005
         Volume state  : ACTIVE
         Size          : 0x017FB800 (12279 MB)
```

```
        GUID          : b5f4a806-758d-11dd-b7f0-c297f0108
        Drive Hint    : E:
```

VolumeID

Windows предоставляет несколько интерфейсов для изменения метки тома, но не дает средств для изменения идентификатора тома, восьмизначного шестнадцатеричного значения, которое отображается в информации о каталогах как серийный номер тома (Volume Serial Number):

```
C:\>dir
  Volume in drive C has no label.
  Volume Serial  Number is 48A6-8C4B
[...]
```

VolumeID — консольная утилита, позволяющая изменять идентификатор томов FAT и NTFS, включая флэш-накопители. VolumeID работает во всех версиях Windows и использует следующий синтаксис:

volumeid *d: xxxx-xxxx*

d — буква диска, а xxxx-xxxx — новое восьмизначное шестнадцатеричное значение идентификатора. На рис. 12-14 показан пример изменения идентификатора диска E на DAD5-1337.

Рис. 12-14. VolumeID

Измененные идентификаторы дисков FAT вступают в силу незамедлительно; для дисков NTFS требуется перезагрузка. С томами exFAT VolumeID не работает.

VolumeID требует разрешения Запись (Write) на целевом томе, которое часто предоставляется только администраторам. Подробнее см. выше во врезке «Разрешения для работы с томами».

Глава 13

Сетевые и коммуникационные утилиты

В этой главе рассказывается об утилитах для работы с сетью и подключениями. TCPView — GUI-версия стандартной утилиты Windows Netstat — показывает конечные точки TCP- и UDP-подключений системы. Whois — утилита командной строки для просмотра информации о регистрации доменов Интернета и поиска DNS-имен по IP-адресам. Portmon — утилита для мониторинга ввода-вывода через последовательные и параллельные порты в реальном времени. В этой главе не рассматриваются утилиты Process Explorer и Process Monitor, хотя обе они имеют функции мониторинга сети (см. гл. 3 и 4).

TCPView

TCPView — программа с графическим интерфейсом, которая показывает и обновляет подробные сведения обо всех конечных точках TCP- и UDP-подключений системы, включая подключения по протоколам IPv4 и IPv6 (рис. 13-1). Для каждой конечной точки выводится имя и идентификатор (PID) процесса-владельца, локальные и удаленные адреса и порты, а также состояния подключений TCP. При запуске с администраторскими правами TCPView показывает и число пакетов, отправленных и принятых через конечные точки. Можно щелкнуть по заголовку любого столбца, чтобы отсортировать данные по этому столбцу.

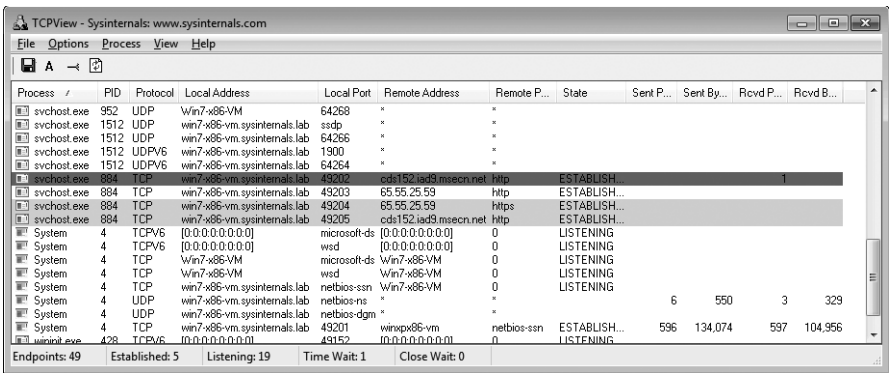


Рис. 13-1. TCPView

По умолчанию экран автоматически обновляется каждую секунду. В меню **View (Вид)** можно выбрать частоту обновления (раз в две или пять секунд) или отключить автоматическое обновление. Для переключения между автоматическим и ручным режимом обновления нажмите пробел. Чтобы вручную обновить окно, нажмите F5. Новые конечные точки, полученные со времени предыдущего обновления, будут выделены зеленым цветом, а конечные точки, которые были удалены с этого времени, — красным. Конечные точки, изменившие состояние, будут выделены желтым.

Параметр **Resolve Addresses (Разрешение адресов)**, включенный по умолчанию, позволяет просматривать доменные имена по IP-адресам и имена служб по номерам портов. Например, порт 445 отображается как «microsoft-ds», а порт 443 как «https». Если отключить этот параметр, будут отображаться только IP-адреса и номера портов. Переключать этот параметр можно нажатием Ctrl+R или кнопкой «A» на панели инструментов. При переключении данные не обновляются.

По умолчанию TCPView показывает все конечные точки. Чтобы отображались только подключенные конечные точки, отключите параметр **Show Unconnected Endpoints (Показывать неподключенные конечные точки)** в меню **Options (Параметры)** или щелкните на соответствующей кнопке панели инструментов. При переключении этого параметра данные обновляются.

Если удаленный адрес является полным доменным именем, можно попробовать выполнить команду «whois» для поиска информации о регистрации домена, щелкнув подключение правой кнопкой мыши и выбрав в контекстном меню команду **Whois**. Если поиск выполнен успешно, TCPView покажет полученную информацию в отдельном окне (рис. 13-2).

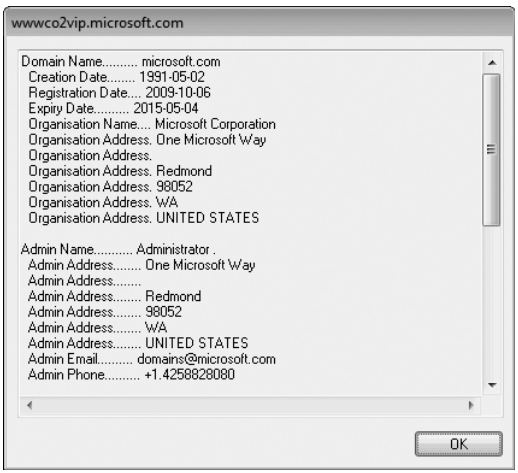


Рис. 13-2. Результаты выполнения команды Whois

Установленное подключение TCP можно закрыть, щелкнув его правой кнопкой мыши и выбрав в контекстном меню команду **Close (Закрыть)**. Эта команда доступна только для подключений TCP IPv4, но не IPv6. Можно

также просмотреть дополнительную информацию о процессе, дважды щелкнув на нем или выбрав команду **Process Properties (Свойства процесса)** из его контекстного меню. Чтобы завершить процесс, выберите в этом меню команду **End Process (Завершить процесс)**.

Чтобы сохранить отображаемые данные в виде текстового файла ASCII с разделителями-знаками табуляции, выберите в меню **File (Файл)** команду **Save (Сохранить)** или **Save As (Сохранить как)**. Можно также копировать данные из одной или нескольких строк в буфер обмена, выделив их и нажав Ctrl+C.

Whois

В большинстве Unix-систем имеется утилита командной строки *whois*, позволяющая искать информацию о регистрации домена и выполнять поиск DNS-имен по IP-адресам. Поскольку в Windows такой утилиты нет, я решил написать ее. Синтаксис прост:

```
whois доменное_имя [сервер-whois]
```

Параметр доменное_имя может указывать имя DNS, такое как sysinternals.com, или адрес IPv4. Можно указать, какой сервер whois должен использоваться для поиска. Если сервер не указан, Whois обращается по стандартному порту whois (TCP 43) к серверу *tld*.whois-servers.net (например, com.whois-servers.net для доменов .com или uk.whois-servers.net для доменов .uk), который отсылает к другим серверам. Whois показывает список опрашиваемых серверов перед полученными данными о регистрации.

Portmon

Portmon записывает все управляющие команды ввода-вывода последовательных и параллельных портов и показывает их в реальном времени вместе с полезной информацией о связанных с ними параметрах. Для запросов чтения и записи Portmon показывает часть прочитанных или записанных данных. По умолчанию эти данные отображаются как символы ASCII (точка представляет непечатаемые символы), но возможен и вывод в шестнадцатеричном виде.

Portmon работает во всех версиях Windows x86 и требует администраторских прав.

Примечание Если запустить Portmon в 64-разрядной версии Windows, будет показана только ошибка «Error 2».

Portmon работает так же, как многие другие мониторы Sysinternals. Сразу после запуска Protmon начинает захватывать команды и данные, отправляемые на все последовательные (COM*n*) и параллельные (LPT*n*) порты, определенные в системе (рис. 13-3).

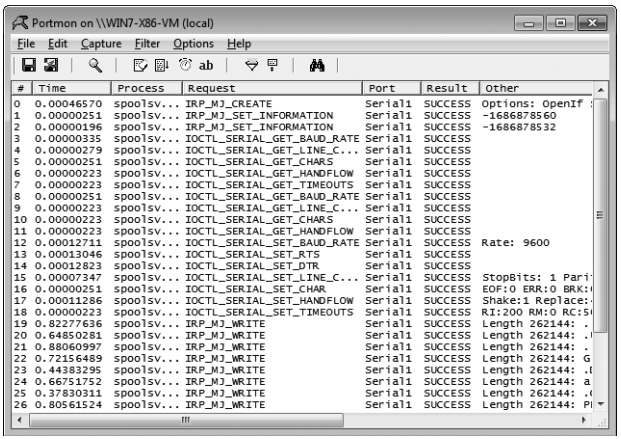


Рис. 13-3. Portmon

Включать и отключать захват данных можно нажатием Ctrl+E или при помощи значка **Capture (Захват)** на панели инструментов. Можно включить автопрокрутку (**Autoscroll**), чтобы при получении новых событий данные прокручивались на экране. Каждое событие представлено отдельной строкой; ширину столбцов можно изменять. Если данные не умещаются в поле, наведите на него указатель мыши — полные данные появятся во всплывающей подсказке.

Первый столбец содержит значение счетчика, сбрасываемое на ноль при очистке экрана. Если данные поступают быстрее, чем Portmon успевает их обработать, или фильтруются (см. ниже), в значениях счетчика могут появляться пропуски. В этом столбце можно отображать время события, если выбрать в меню **Options (Параметры)** команду **Clock Time (Время)**. Изменение этого параметра влияет только на данные, полученные позже. Столбец времени можно скрыть, отключив параметр **Show Time Column (Показывать столбец времени)** в меню **Options (Параметры)**.

В столбце **Process (Процесс)** показано имя процесса, сделавшего запрос.

В столбце **Request (Запрос)** показано символьное имя управляющего кода, отправленного на порт. Большинство имен объясняют значение соответствующих команд (тем, кто сведущ во взаимодействии через порты). IOCTL означает input/output control (управление вводом-выводом), IRP — input/output request packet (пакет запроса ввода-вывода), MJ — major function (основная функция). Команды IOCTL используются для настройки устройств, а команды IRP обычно запрашивают или содержат данные.

В столбце **Port (Порт)** показано имя порта, на который отправлен запрос. По умолчанию Portmon следит за всеми последовательными портами, указанными в разделе HKLM\Hardware\DeviceMap\SerialComm и всеми параллельными портами, указанными в разделе HKLM\Hardware\DeviceMap\Parallel Ports. Можно выборочно отключить наблюдение за

определенными портами, используя меню **Capture | Ports (Захват | Порты)** (рис. 13-4). Portmon запоминает выбор портов и воспроизводит его при следующем запуске.

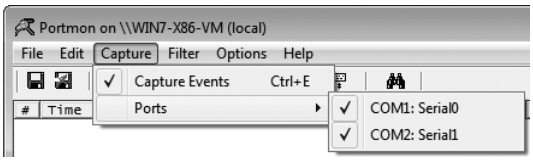


Рис. 13-4. Выбор портов

В столбце **Result (Результат)** показан результат запроса.

В столбце **Other (Другое)** показываются другие данные, имеющие отношение к запросу. Например, для команды IOCTL «set baud rate» (установить скорость передачи) Portmon показывает в этом столбце запрошенную скорость. Для операций чтения и записи Portmon показывает длину данных и, как минимум, часть данных. По умолчанию отображается 64 байта данных в формате ASCII, где точка представляет непечатаемые символы. Объем отображаемых данных можно изменить, выбрав в меню **Options (Параметры)** команду **Max Bytes (Максимум байт)** и указав нужное число в диалоговом окне **Max Bytes (Максимум байт)**. Можно также отображать данные в шестнадцатеричном виде, выбрав в меню **Options (Параметры)** команду **Show Hex (Отображать в шестнадцатеричном виде)**. Оба этих параметра применяются к данным, полученным в дальнейшем. Формат уже показанных данных не меняется.

Portmon следит за использованием системной памяти. Утилита приостанавливает захват данных, если обнаруживает нехватку памяти, и возобновляет, когда появляется свободная память. Один из способов ограничить потребление памяти утилитой Portmon — задать параметру **History Depth (Период журнала)** ненулевое значение. Этот параметр меню **Options (Параметры)** ограничивает число отображаемых событий; остальные события удаляются.

Можно увеличить пространство для отображения выходных данных, выбрав в меню **Options (Параметры)** команду **Hide Toolbar (Скрыть панель инструментов)**. Также можно увеличить число видимых строк, выбрав шрифт меньшего размера. Чтобы изменить шрифт, выберите в меню **Options (Параметры)** команду **Font (Шрифт)**.

В отличие от большинства утилит Sysinternals, хранящих свои параметры в разделе HKCU\Software\Sysinternals, Portmon хранит параметры в разделе HKCU\Software\Systems lnternals\Portmon, за исключением флага EulaAccepted, хранящегося в разделе HKCU\Software\Sysinternals\Portmon.

Поиск, фильтрация и выделение

Если нужно найти строку, содержащую определенный текст, нажмите Ctrl+F, чтобы открыть диалоговое окно **Find (Найти)**. Если введенный текст найден

в окне выходных данных, Portmon выделит строку, содержащую его, и отключит автопрокрутку, чтобы эта строка отображалась в окне. Чтобы повторить поиск, нажмите F3.

Другой способ изолировать интересующие вас выходные данные — использовать фильтрацию. Щелкните кнопку **Filter (Фильтр)** на панели инструментов Portmon, чтобы открыть диалоговое окно **Filter (Фильтр)** (рис. 13-5). Правила фильтрации и выделения сохраняются автоматически и могут применяться при следующем запуске Portmon.

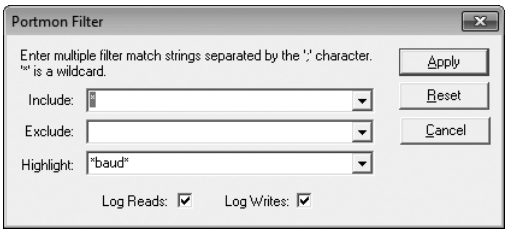


Рис. 13-5. Диалоговое окно Portmon Filter

В поле **Include (Включить)** введите выражения, соответствующие содержимому столбцов, которое вы хотите отображать. В поле **Exclude (Исключить)** укажите содержимое, строки с которым не должны отображаться. В поле **Highlight (Выделить)** укажите, какое содержимое должно выделяться цветом. Можно водить несколько выражений, разделяя их точками с запятой. Не вводите пробелы в выражения фильтра, если их не должно быть в фильтре. Символ «*» интерпретируется как знак подстановки. Выражения фильтра нечувствительны к регистру. По умолчанию включено все («*»), и ничего не исключено.

Чтобы правило применялось, выражение должно полностью соответствовать значению столбца. Например, если хотите указать строку, содержащую слово «baud» между другим текстом (например, IOCTL_SERIAL_GET_BAUD_RATE), добавьте знаки подстановки перед этим словом и после него: *baud*.

Приведем пример. Допустим, вы хотите отображать все события, кроме имеющих поле, оканчивающееся текстом «_WRITE», и выделять события, имеющие слово «baud». Фильтр будет таким:

```
Include:   *
Exclude:   *_write
Highlight:   *baud*
```

Фильтрация применяется только к новым событиям при их получении. Новые строки, соответствующие указанным правилам, будут отображаться. Отброшенные фильтром строки не «вернутся», а уже показанные — не удалятся, если изменить правила фильтрации.

Выделение (подсветка) применяется сразу после определения правил фильтрации. По умолчанию для подсветки используется белый текст на

красном фоне. Изменить эти цвета можно, выбрав в меню **Filter (Фильтр)** команду **Highlight Colors (Цвета выделения)**.

Если при выходе из Portmon действовали какие-то правила включения или исключения, они будут показаны в отдельном окне при следующем запуске. Просто щелкните **OK**, чтобы продолжить их использование, или измените их. Можно щелкнуть кнопку **Reset (Сброс)**, чтобы удалить фильтр.

Сохранение, запись в журнал и печать

Можно выделить одну или несколько строк из окна Portmon и скопировать их в буфер обмена в виде текста с разделителями-табуляторами, нажав Ctrl+C. Столбец **Time (Время)** копируется всегда, даже если отключен параметр **Show Time Column (Показывать столбец времени)**, а порядковый номер не копируется. Portmon поддерживает стандартные способы выделения нескольких строк, используемые в Windows: чтобы выделить несколько смежных строк, удерживайте Shift и нажимайте клавиши со стрелками вверх или вниз; чтобы выделить несколько несмежных строк, удерживайте клавишу Ctrl и щелкните нужные строки.

Можно сохранить полученные данные в текстовом файле, выбрав в меню **File (Файл)** команду **Save (Сохранить)** или **Save As (Сохранить как)**. По умолчанию Protmon использует расширение .LOG. Текст сохраняется в формате ASCII с разделением табуляцией и всегда включает порядковые номера и столбец времени.

Чтобы данные записывались в файл одновременно с отображением на экране, выберите в меню **File (Файл)** команду **Log To File (Запись в файл)**. При первом выборе этой команды или первом нажатии кнопки **Log To File** на панели инструментов открывается окно **Log-To-File Settings (Параметры записи в файл)** для выбора файла, в который будет вестись запись (рис. 13-6). После этого при включении записи данные будут записываться в этот файл. Чтобы выбрать другой файл или изменить параметры записи, выберите в меню **File (Файл)** команду **Log To File As (Запись в виде...)**. (Если запись в файл включена, эта команда отключает ее.) В диалоговом окне параметров записи можно указать максимальный размер файла и способ записи данных: перезапись файла или добавление. Когда файл достигает указанного размера, запись в файл отключается, но данные продолжают выводиться на экран. Значение 0 указывает, что ограничений на размер файла нет.

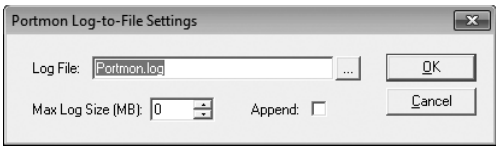


Рис. 13-6. Диалоговое окно Portmon Log-to-File Settings

При записи в файл Portmon записывает для каждого события две строки выходных данных с одним порядковым номером. В первой строке указан

процесс, запрос, порт и другую информацию, предоставленную процессом. Во второй строке указывается длительность события (если не включен параметр **Clock Time**) и результат. Строки, относящиеся к одному событию, могут идти не подряд, если операция выполняется асинхронно. Столбцы разделяются двумя пробелами, а не табуляцией. Вот пример журнала Portmon:

```
0  0.00000000  spoolsv.exe  IRP_MJ_CREATE  Serial1  Options:  OpenIf Sequential
0  0.00065288  SUCCESS
1  0.00000000  spoolsv.exe  IRP_MJ_SET_INFORMATION  Serial1  -1699359072
1  0.00000587  SUCCESS
2  0.00000000  spoolsv.exe  IRP_MJ_SET_INFORMATION  Serial1  -1699359044
2  0.00000223  SUCCESS
3  0.00000000  spoolsv.exe  IOCTL_SERIAL_GET_BAUD_RATE  Serial1
3  0.00000615  SUCCESS
4  0.00000000  spoolsv.exe  IOCTL_SERIAL_GET_LINE_CONTROL  Serial1
4  0.00000279  SUCCESS
```

Чтобы распечатать полученные данные, выберите в меню файл команду **Print (Печать)** или **Print Range (Печать диапазона)**. Первая команда выводит на печать все записи, вторая позволяет распечатать только часть строк с указанными порядковыми номерами. Перед выводом на печать необходимо отключить захват данных.

В диалоговом окне **Print Range (Печать диапазона)** можно также указать, следует ли печатать порядковые номера и столбец времени (рис. 13-7). Эти поля можно не печатать, чтобы сэкономить бумагу. Установленные параметры используются во всех последующих операциях печати.

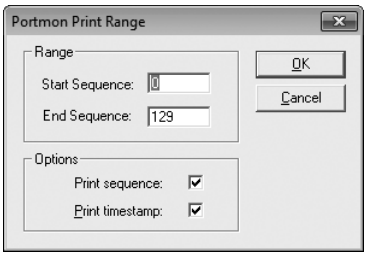


Рис. 13-7. Диалоговое окно Portmon Print Range

Чтобы не переносить длинные строки, используйте альбомную ориентацию страницы.

Глава 14

Утилиты для получения информации о системе

В этой главе рассматриваются утилиты, отображающие сведения о системе, но не попадающие в категории, описанные в предыдущих главах:

- **RAMMap** предоставляет подробные сведения о выделении физической памяти;
- **CoreInfo** показывает сопоставления логических процессоров с физическими, узлами NUMA и гнездами, в котором они установлены, а также кеши логических процессоров и стоимость взаимодействия между узлами в системах NUMA;
- **ProcFeatures** сообщает, поддерживает ли процессор и текущая версия Windows различные функции, такие как страницы памяти No-Execute;
- **WinObj** обеспечивает навигацию по пространству имен диспетчера объектов Windows и просмотр информации о содержащихся в нем объектах;
- **LoadOrder** показывает примерный порядок, в котором Windows загружает драйверы устройств и запускает службы;
- **PipeList** показывает список именованных каналов локального компьютера;
- **ClockRes** показывает текущее разрешение системных часов.

RAMMap

RAMMap — утилита для анализа использования физической памяти, показывающая, как Windows распределяет физическую память, также называемую памятью с произвольным доступом (random access memory, RAM). RAMMap предоставляет информацию об использовании памяти, исходя из различных параметров, включая тип использования, список страниц, процессы, файлы, приоритеты и физические адреса. При помощи RAMMap можно также очищать отдельные области памяти. Это удобно, когда требуется проверить корректность управления памятью, используя чистую область памяти в качестве отправной точки. Наконец, RAMMap поддерживает сохранение и загрузку моментальных снимков памяти. RAMMap работает в Windows Vista и выше, требует администраторских прав.

Все процессы пользовательского режима и большинство программ режима ядра обращается к коду и данным через адреса виртуальной памяти. Этот код и данные могут находиться в физической памяти или в файле на диске, но должны быть спроецированы в рабочий набор процесса[1] — физическую память, которую диспетчер памяти назначает процессу, — когда процесс считывает и записывает данные или выполняет код. Утилита VMMap, описанная в главе 7, показывает память с точки зрения виртуального адресного пространства процесса: сколько занято исполняемыми и другими проецированными файлами; сколько занято стеками, кучами и другими областями данных; сколько виртуальной памяти спроецировано в рабочий набор процесса; сколько памяти не используется. RAMMap представляет память как общесистемный ресурс, совместно используемый всеми процессами. Виртуальная память процесса, выделенная и разделенная на страницы, в RAMMap не отображается. На рис. 14-1 показана вкладка Use Counts (Счетчики использования) окна RAMMap.

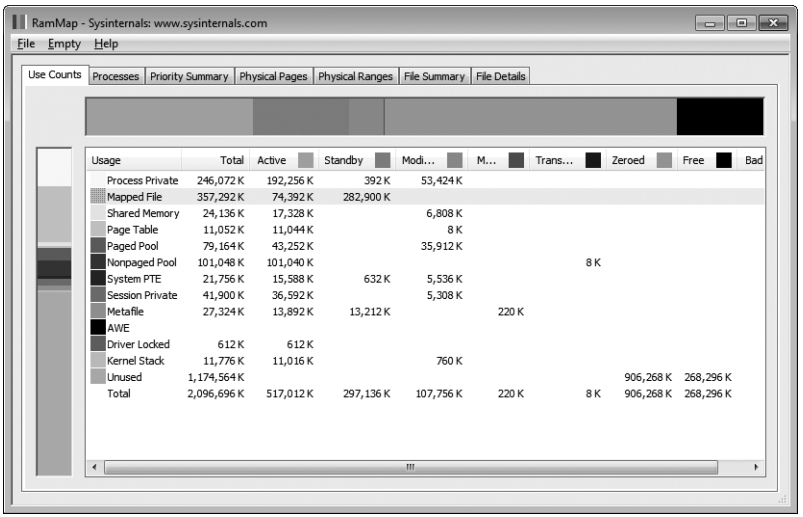


Рис. 14-1. Вкладка Use Counts в RAMMap

На семи вкладках RAMMap показаны данные о памяти, представленные с различных позиций, включая тип, список страниц, использование процессами, приоритет, проецируемые файлы и т.д. Некоторые вкладки содержат очень много информации. Чтобы быстро найти строку, содержащую искомый текст, такой как имя файла или процесса, нажмите Ctrl+F, чтобы открыть окно **Find (Найти)**, или нажмите F3, чтобы повторить предыдущий поиск. Данные можно обновить в любое время, нажав F5.

Подробнее об описанных здесь концепциях см. в пятой редакции книги «Windows Internals» (Microsoft Press, 2009).

---

[1] Обычно это так, но не всегда: расширения оконного доступа к адресам (Address Windowing Extension, AWE) и крупные страницы памяти не входят в рабочий набор, даже при обращении к ним.

Счетчики памяти

Таблицы и диаграммы на вкладке **Use Counts (Счетчики)** показывают данные об использовании разных типов и страниц памяти (рис. 14-1). В столбцах таблицы и на итоговой диаграмме над таблицей показано, сколько оперативной памяти находится в каждом из списков диспетчера памяти. В строках таблицы и на сводной диаграмме слева от нее показано распределение оперативной памяти по типам. Цвета заголовков строк и столбцов служат ключом для интерпретации соответствующих диаграмм. Порядок столбцов можно изменять, перетаскивая их за заголовки, и можно сортировать таблицу по содержимому столбцов, щелкая заголовки этих столбцов. Последовательные щелчки заголовков переключают сортировку по возрастанию и по убыванию.

На этой вкладке показаны следующие списки страниц памяти:

- **Active (Активная)** — память, которая доступная для обращения без обработки страничной ошибки. Сюда включена память, находящаяся в рабочем наборе одного или нескольких процессов или одном из системных рабочих наборов (например, рабочем наборе системного кеша), а также неподкачиваемая память, такая как пул неподкачиваемой памяти и AWE.
- **Standby (Ожидающая)** — кешированная память, которая удалена из рабочего набора, но может вернуться в него (в активную память) после обработки «мягкой» страничной ошибки (soft fault). К этой памяти можно обратиться без дискового ввода-вывода.
- **Modified (Измененная)** — память, удаленная из рабочего набора, модифицированная, но еще не записанная на диск. Ее можно вернуть в рабочий набор после «мягкой» страничной ошибки, но перед повторным использованием она должна быть записана на диск.
- **Modified no write (Измененная, без записи)** — то же, что и **Modified**, за тем исключением, что по запросу драйверов файловой системы страница была помечена как не записываемая на диск автоматически, например, из за записи транзакций NTFS.
- **Transition (Переход)** — временное состояние страницы, заблокированной в памяти драйвером для выполнении ввода-вывода.
- **Zeroed (Обнуленная)** — память, заполненная нулями и доступная для выделения.
- **Free (Свободная)** — незанятая и не обнуленная память. Доступна для выделения кодом режима ядра и пользовательского режима после инициализации. При необходимости диспетчер памяти может обнулять страницы свободной памяти перед передачей их пользовательскому процессу. Поток обнуления страниц, запускаемый с более низким приоритетом, чем все остальные потоки, заполняет свободные страницы нулями и перемещает их в список **Zeroed (Обнуленная)**, из-за чего в этом списке обычно очень мало страниц.

- **Bad (Плохая)** — непригодная к использованию память, сгенерировавшая ошибку контроля четности или другую аппаратную ошибку. В этот же список Windows помещает страницы, переходящие из одного состояние в другое, и внутренние списки страниц.

В строках таблицы отображаются следующие типы памяти:

- **Process Private (Закрытая память процесса)** — память, которая может использоваться только одним процессом.
- **Mapped File (Проецируемый файл)** — разделяемая память, представляющая содержимое файла на диске. Примеры проецируемых файлов — исполняемые образы и DLL ресурсов.
- **Shared Memory (Разделяемая память)** — память, которая может сообща использоваться несколькими процессами и выгружаться в файл подкачки.
- **Page Table (Страничная таблица)** — память режима ядра с пространствами виртуальных адресов процесса.
- **Paged Pool (Выгружаемый пул)** — выделяемая ядром память, которую можно выгрузить на диск.
- **Nonpaged Pool (Невыгружаемый пул)** — выделяемая ядром память, которая должна всегда оставаться в физической памяти. Память из этого пула фигурирует только в столбце **Active (Активная)**.
- **System PTE (Системные PTE)** — память, содержащая записи системной страничной таблицы (page table entry, PTE), которые используются для динамического проецирования системных страниц, таких как диапазоны ввода-вывода, стеки ядра и проекции списков описателей памяти.
- **Session Private (Закрытая память сеанса)** — память, выделяемая кодом Win32k.sys или сеансовыми драйверами (например, драйверами видеоадаптера, клавиатуры и мыши) для использования одним сеансом служб терминалов.
- **Metafile (Метафайл)** — память для метаданных файловой системы, включая каталоги, файлы подкачки и файлы метаданных NTFS, таких как MFT.
- **AWE** — память, используемая расширениями AWE. AWE — это набор функций, с помощью которых программы могут управлять данными, хранящимися в оперативной памяти.
- **Driver Locked (Блокированная драйверами)** — память, выделяемая драйверами из системного пула памяти, всегда находится в списке активных страниц. Hyper-V и Virtual PC используют такую память в качестве оперативной памяти виртуальных машин.
- **Kernel Stack (Стек ядра)** — память стеков потоков режима ядра.
- **Unused (Неиспользуемая)** — неиспользуемая память; всегда находится в списках **Zeroed**, **Free** или **Bad**.

Процессы

На вкладке **Processes (Процессы)** показано разделение по категориям страниц физической памяти, которые могут быть связаны с одним процессом (рис. 14-2). Сюда включена закрытая память процессов пользовательского режима, а также память режима ядра, содержащая таблицы страниц процессов. В столбцах **Private**, **Standby** и **Modified** показан объем закрытой памяти процесса из списков **Active**, **Standby** и **Modified**, соответственно. В столбце **Page Table** показана суммарная память режима ядра, выделенная под таблицы страниц процессов.

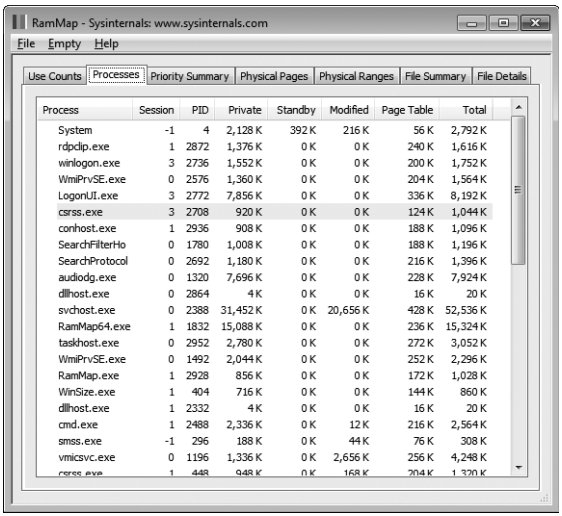


Рис. 14-2. RAMMap, вкладка Processes

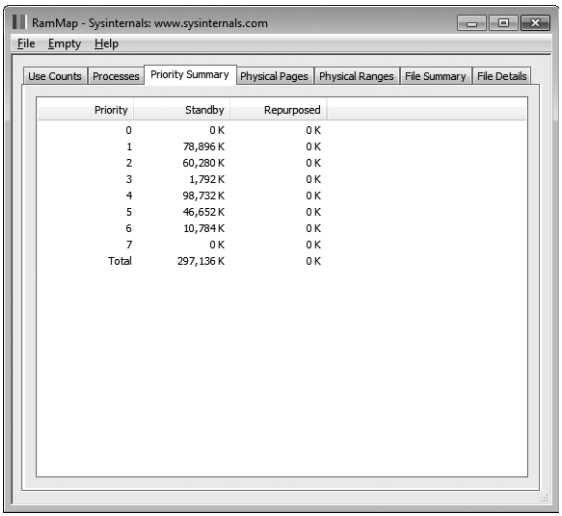


Рис. 14-3. RAMMap, вкладка Priority Summary

Сводка по приоритетам

На вкладке **Priority Summary (Сводка по приоритетам)** показана память, которая в данный момент числится в списках ожидания, с разделением по приоритетам (рис. 14-3). В столбце **Repurposed (Изменено назначение)** показана память, взятая с момента запуска системы из списков ожидания для удовлетворения новых запросов выделения памяти, а не возвращенная в прежний рабочий набор после «мягкой» страничной ошибки. Частое изменение назначения для приоритетов от 5 и выше может говорить о том, что система испытывала или испытывает нехватку памяти, ее производительность можно повысить путем наращивания оперативной памяти.

Физические страницы

На вкладке **Physical Pages (Физические страницы)** показана информация о памяти на уровне отдельных страниц в следующих столбцах:

Physical Address (Физический адрес) — физический адрес страницы;

- **List (Список)** — список, в котором числится страница;
- **Use (Назначение)** — тип, например **Process Private (Закрытая процесса)**, **Kernel Stack (Стек режима ядра)** или **Unused (Неиспользуемая)**;
- **Priority (Приоритет)** — текущий приоритет страницы;
- **Image (Образ)** — содержит метку «Yes» («Да»), если страница содержит проецируемый файл образа или его часть;
- **Offset (Смещение)** — указывает смещение в страничной таблице или проецируемом файле, которое представляет страница;
- **File Name (Имя файла)** — имя проецируемого файла, данные которого содержит физическая страница;
- **Process (Процесс)** — процесс-владелец (если память связана с одним процессом);
- **Virtual Address (Виртуальный адрес)** — для памяти типа **Process Private (Закрытая память процесса)** это виртуальный адрес в адресном пространстве процесса, для памяти режима ядра, такой как **System PTE**, — виртуальный адрес в системном пространстве;
- **Pool Tag (Метка пула)** — метка выгружаемого или невыгружаемого пулов (если есть). Имеется только у страниц, расположены целиком в одной выделенной области.

Два раскрывающихся списка в нижней части вкладки позволяют фильтровать физические страницы, отображаемые в таблице. В первом раскрывающемся списке выберите столбец, по которому нужно отфильтровать данные, а во втором — значение. Дальнейший анализ можно облегчить, щелкнув заголовок столбца, чтобы отсортировать фильтрованные результаты. Например, чтобы отображались только страницы с приоритетом 7, выберите **Priority (Приоритет)** в первом раскрывающемся списке, а во втором выберите **7**. Щелкните на заголовке столбца **Use (Использование)**, чтобы

было проще увидеть, каким типам распределения назначен приоритет 7 (рис. 14-4).

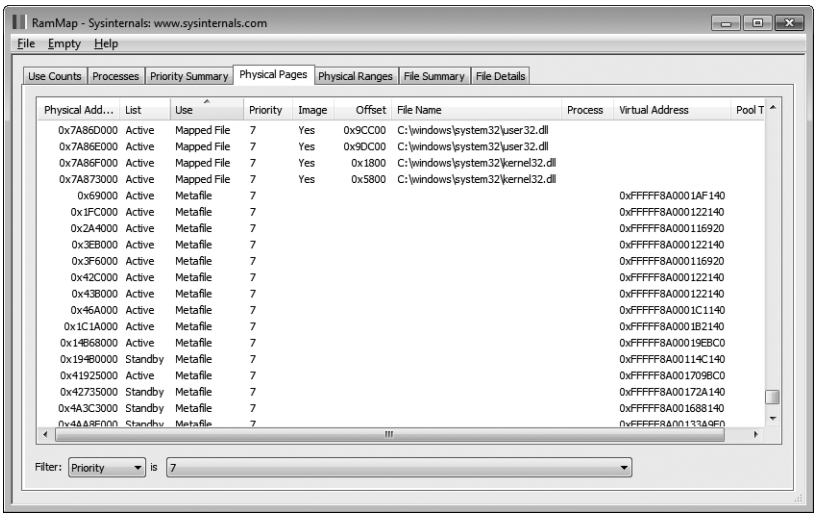


Рис. 14-4. RAMMap, вкладка Physical Pages

Физические диапазоны

Вкладка **Physical Ranges (Физические диапазоны)** содержит списки действительных диапазонов адресов физической памяти (рис. 14-5). Пропуски в нумерации обычно указывают на физические адреса, выделенные устройствам.

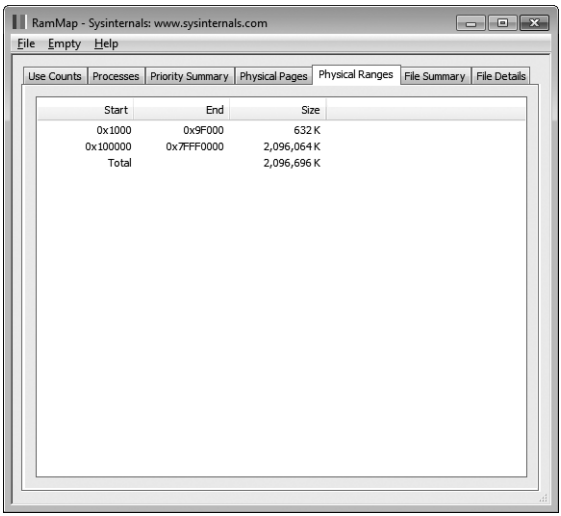


Рис. 14-5. RAMMap, вкладка Physical Ranges

Сводка по файлам

На вкладке **File Summary (Сводка по файлам)** показаны пути ко всем проецируемым файлам, имеющим данные в оперативной памяти (рис. 14-6). Для каждого файла показан общий объем занимаемой оперативной памяти, объем активной памяти (находящейся в одном или нескольких рабочих наборах) и объем памяти, находящейся в списках **Standby**, **Modified** и **Modified No-Write**. Как и в других таблицах RAMMap, столбцы можно сортировать, щелкая заголовки, и перемещать.

Windows может проецировать файлы в память по следующим причинам:

- исполняемые файлы и библиотеки DLL проецируются загрузчиком, когда загружаются для выполнения;
- приложение может явно проецировать файл вызовом API-функции *MapViewOfFile*;
- диспетчер кеша может проецировать файл, когда приложение выполняет с ним кешируемую операцию ввода-вывода;
- служба Superfetch выполняет упреждающее внесение исполняемых и других файлов в список ожидания.

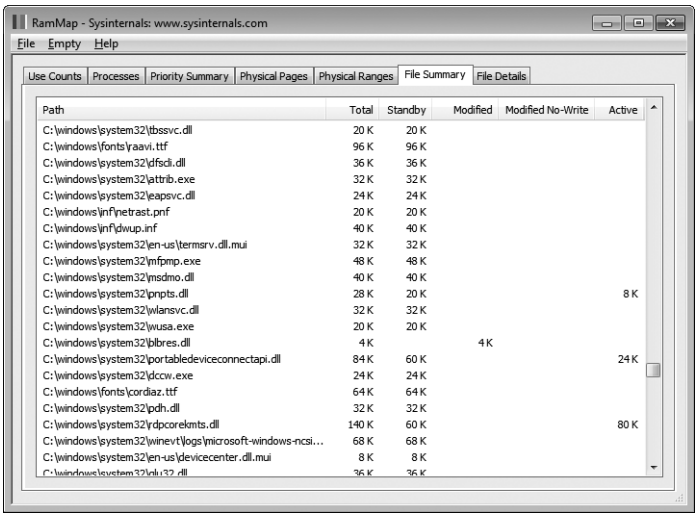


Рис. 14-6. RAMMap, вкладка File Summary

Сведения о файле

На вкладке **File Details (Сведения о файле)**, как и на вкладке **File Summary (Сводка по файлам)**, показаны пути ко всем проецируемым файлам, имеющим данные в оперативной памяти, и общий объем памяти, занимаемой каждым файлом. Если щелкнуть знак + рядом с именем файла, раскроется список всех физических страниц, занятых этим файлом. Для каждой страницы показан физический адрес, список, в который включена страница,

тип (это всегда **Mapped File**), приоритет памяти, смещение в проецируемом файле, которое представляет страница, а также указано, загружена ли страница как исполняемый образ.

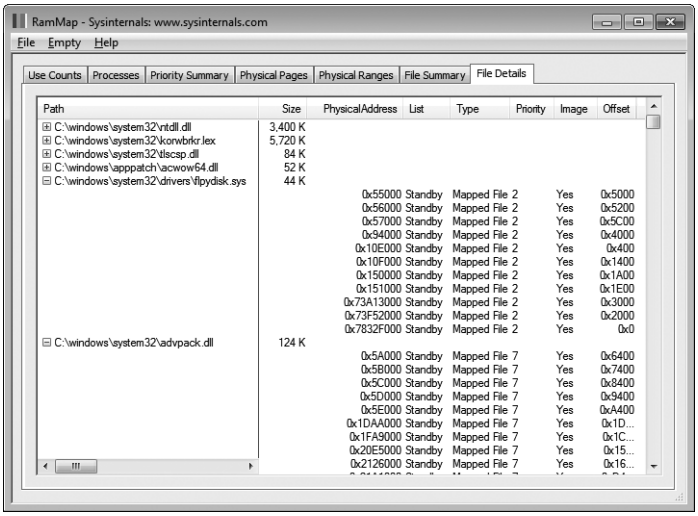


Рис. 14-7. RASMMap, вкладка File Details

Очистка физической памяти

RAMMap дает возможность очищать рабочие наборы и списки страниц. Это удобно для оценки использования памяти приложениями и их отдельными функциями. Например, можно сравнивать влияние различных функций на физическую память, очистив все рабочие наборы перед выполнением функции и получив снимок экрана после ее выполнения.

Выберите в меню **Empty (Очистить)** одну из команд, описанных ниже, и RAMMap незамедлительно очистит соответствующую область памяти. Заметьте, что RAMMap не обновляет данные автоматически, так что можно очистить несколько областей памяти, а потом нажать F5, чтобы обновить окно RAMMap.

- **Empty Working Sets (Очистить рабочие наборы)** — удаляет память из всех рабочих наборов пользовательского режима и системы, перенося ее страницы в списки **Standby** и **Modified**;
- **Empty System Working Sets (Очистить системные рабочие наборы)** — удаляет память из рабочего набора системного кеша;
- **Empty Modified Page List (Очистить список измененных страниц)** — очищает память из списка страниц **Modified**, записывая несохраненные данные на диск и перенося страницы в список **Standby**;
- **Empty Standby List (Очистить список ожидания)** — удаляет страницы из всех списков **Standby** и переносит их в список **Free**;

- **Empty Priority 0 Standby List (Очистить список ожидания с приоритетом 0)** — переносит страницы из списка **Standby** с низшим приоритетом в список **Free**.

Сохранение и загрузка моментальных снимков

Всю информацию о моментальном снимке RAMMap можно сохранить для дальнейшего просмотра (на том же или другом компьютере), выбрав команду меню **File | Save (Файл | Сохранить)**. По умолчанию RAMMap использует для своих файлов расширение .RMP, но не создает такую ассоциацию. Моментальные снимки RAMMap — XML-файлы с частично кодированным содержимым. Чтобы просмотреть сохраненный моментальный снимок, выберите команду меню **File | Open (Файл | Открыть)** и укажите файл снимка.

CoreInfo

CoreInfo — утилита командной строки, которая показывает сопоставления логических процессоров физическим, узлам NUMA, группам процессоров (в Windows 7 и выше) и гнездам, а также кеши логических процессоров. Она получает эту информацию при помощи функции Windows *GetLogicalProcessorInformation* или *GetLogicalProcessorInformationEx*, в зависимости от версии ОС, и выводит данные на экран, представляя связь с логическим процессором звездочкой. Кроме получения информации о топологии NUMA, эта утилита оценивает и показывает стоимость взаимодействия узлов в системах NUMA. CoreInfo позволяет получить представление о топологии процессоров и кешей системы.

CoreInfo работает в Windows XP с третьим пакетом обновлений и выше, а также в Windows Server 2003 и выше, включая системы IA-64. Администраторские права не требуются.

При запуске без параметров командной строки CoreInfo выводит всю собранную информацию. Чтобы вывести только часть сведений, используйте следующие параметры командной строки:

- -c — выводит информацию о ядрах;
- -g — выводит информацию о группах;
- -l — выводит информацию о кешах;
- -n — выводит информацию об узлах NUMA;
- -s — выводит информацию о сокетах;
- -m — выводит информацию о стоимости взаимодействия NUMA.

Ниже приведен пример вывода для четырехядерной системе AMD Opteron с двумя гнездами ЦП. Заметьте, что каждое гнездо сопоставлено узлу NUMA, а каждый ЦП имеет собственную инструкцию и кеш данных первого уровня, унифицированный кеш второго уровня и разделяемый унифицированный кеш третьего уровня.

```
Coreinfo v2.11 - Dump information on system CPU and  memory topology
Copyright  (C)  2008-2010 Mark Russinovich Sysinternals  - www.sysinternals.com
```

```
Logical to Physical Processor Map:
*-------  Physical Processor 0
-*------  Physical Processor 1
--*-----  Physical Processor 2
---*----  Physical Processor 3
----*---  Physical Processor 4
-----*--  Physical Processor 5
------*-  Physical Processor 6
-------*  Physical Processor 7

Logical Processor to Socket Map:
********  Socket 0
--------  Socket 1

Logical Processor to NUMA Node Map:
********  NUMA Node 0
--------  NUMA Node 1

Approximate Cross-NUMA Node Access Cost (relative to fastest):
     00 01
00: 1.0 1.4
01: 1.5 1.2

Logical Processor to Cache Map:
*-------  Data Cache          0, Level 1,   64 KB, Assoc   2, LineSize  64
*-------  Instruction Cache   0, Level 1,   64 KB, Assoc   2, LineSize  64
*-------  Unified Cache       0, Level 2,  512 KB, Assoc  16, LineSize  64
-*------  Data Cache          1, Level 1,   64 KB, Assoc   2, LineSize  64
-*------  Instruction Cache   1, Level 1,   64 KB, Assoc   2, LineSize  64
-*------  Unified Cache       1, Level 2,  512 KB, Assoc  16, LineSize  64
--*-----  Data Cache          2, Level 1,   64 KB, Assoc   2, LineSize  64
--*-----  Instruction Cache   2, Level 1,   64 KB, Assoc   2, LineSize  64
--*-----  Unified Cache       2, Level 2,  512 KB, Assoc  16, LineSize  64
---*----  Data Cache          3, Level 1,   64 KB, Assoc   2, LineSize  64
---*----  Instruction Cache   3, Level 1,   64 KB, Assoc   2, LineSize  64
---*----  Unified Cache       3, Level 2,  512 KB, Assoc  16, LineSize  64
********  Unified Cache       4, Level 3,    2 MB, Assoc   1, LineSize  64
----*---  Data Cache          4, Level 1,   64 KB, Assoc   2, LineSize  64
----*---  Instruction Cache   4, Level 1,   64 KB, Assoc   2, LineSize  64
----*---  Unified Cache       5, Level 2,  512 KB, Assoc  16, LineSize  64
-----*--  Data Cache          5, Level 1,   64 KB, Assoc   2, LineSize  64
-----*--  Instruction Cache   5, Level 1,   64 KB, Assoc   2, LineSize  64
-----*--  Unified Cache       6, Level 2,  512 KB, Assoc  16, LineSize  64
------*-  Data Cache          6, Level 1,   64 KB, Assoc   2, LineSize  64
------*-  Instruction Cache   6, Level 1,   64 KB, Assoc   2, LineSize  64
------*-  Unified Cache       7, Level 2,  512 KB, Assoc  16, LineSize  64
-------*  Data Cache          7, Level 1,   64 KB, Assoc   2, LineSize  64
```

```
-------*  Instruction Cache   7, Level 1,    64 KB, Assoc   2, LineSize  64
-------*  Unified Cache       8, Level 2,   512 KB, Assoc  16, LineSize  64
---*****  Unified Cache       9, Level 3,     2 MB, Assoc   1, LineSize  64

Logical Processor to Group Map:
********  Group 0
```

ProcFeatures

ProcFeatures — простая утилита командной строки, определяющая с помощью функции Windows *IsProcessorFeaturePresent*, поддерживает ли процессор и Windows различные функции, такие как страницы памяти No-Execute, расширения физических адресов (Physical Address Extensions, PAE) и счетчик циклов реального времени. Основная задача этой утилиты — выявление систем с ядром PAE и поддерживающих аппаратный запрет выполнения данных (Data Execution Prevention). ProcFeatures не требует администраторских прав. На рис. 14-8 показан пример выходных данных для системы Intel Core2 Duo с 64-разрядной версией Windows 7.

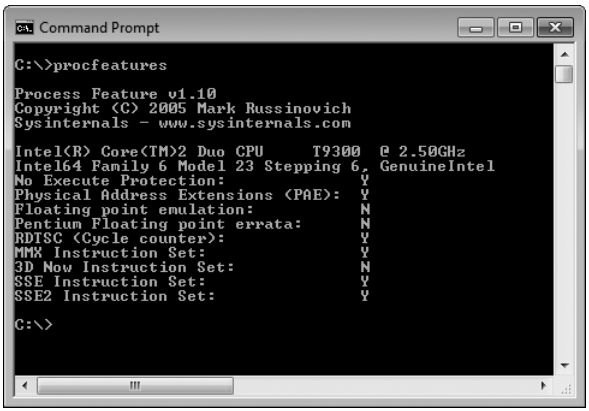


Рис. 14-8. ProcFeatures

WinObj

WinObj — утилита с графическим интерфейсом, которая позволяет просматривать пространство имен диспетчера объектов Windows и информацию о содержащихся в нем объектах. Диспетчер объектов поддерживает иерархию каталогов и общий интерфейс для создания, удаления и защиты объектов и обращения к ним. Подробнее о Диспетчере объектов см. в пятой редакции книги «Windows Internals».

WinObj работает во всех версиях Windows и не требует администраторских прав. Однако при запуске с администраторскими правами можно получить больше информации, так как некоторые области пространства имен Диспетчера объектов требуют их даже для просмотра. А поскольку доступ

к некоторым объектам есть только у системной учетной записи, для получения наиболее полных данных обычно нужно запустить WinObj от имени системы (это можно сделать при помощи утилиты PsExec, описанной в главе 6). В Windows Vista и выше можно перезапустить WinObj с повышенными правами, выбрав команду **File | Run As Administrator (Файл | Запуск от имени администратора)**. Как показано на рис. 14-9, WinObj отображает иерархию каталогов Диспетчера объектов в виде дерева в левой части окна. Имя корневого каталога — просто косая черта. Если выбрать каталог в левой части окна, в правой части будут показаны объекты, содержащиеся в этом каталоге. Если выбрать каталог в левой части или объект в правой, в строке состояния будет показан полный путь к элементу. Данные можно обновить в любое время, нажав F5.

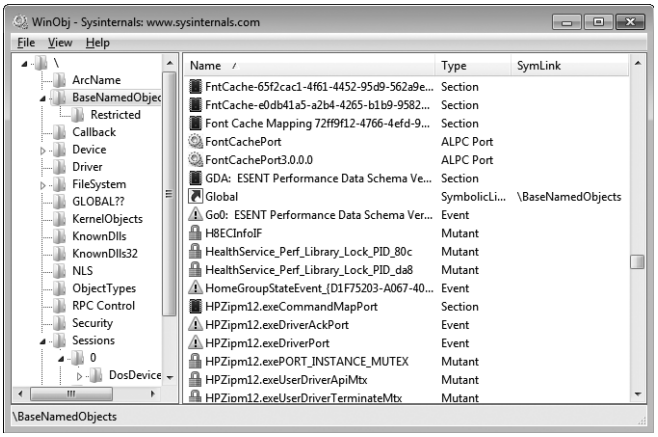


Рис. 14-9. WinObj

Сортируемая таблица в правой части окна содержит имена и типы объектов; для символьных ссылок в столбце **SymLink** указана цель ссылки. Чтобы отсортировать список объектов по какому-либо столбцу, щелкните на его заголовке. Рядом с именем объекта показан значок, соответствующий типу объекта:

- мьютексы (мутанты) — замок;
- разделы (объекты-проекции файлов) — чип памяти;
- события — треугольник с восклицательным знаком;
- события KeyedEvents — тот же значок, что и у событий, но с ключом;
- семафоры — значок, напоминающий дорожный знак;
- символьные ссылки — изогнутая стрелка;
- устройства — изображение компьютера;
- драйверы — стандартный значок .sys-файлов;
- оконные станции — монитор;
- таймеры — часы;
- другие объекты, такие как порты и задачи ALPC — шестеренки.

Для просмотра дополнительной информации о каталоге или объекте щелкните правой кнопкой мыши и выберите команду **Properties (Свойства)**. Окно свойств (рис. 14-10) можно также открыть двойным щелчком на объекте (но если это символьная ссылка, будет выполнен переход по ссылке).

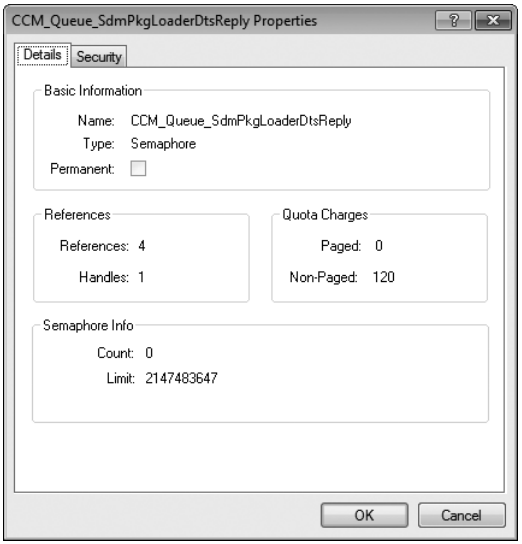


Рис. 14-10. Окно свойств объекта

Вкладка **Details (Подробно)** диалогового окна свойств содержит следующую информацию для объектов всех типов:

- имя и тип объекта;
- является ли объект «постоянным», т.е. объектом, который не удаляется автоматически, когда ссылок на него не остается;
- счетчики ссылок и описателей. Поскольку каждый описатель включает ссылку на объект, счетчик ссылок никогда не бывает меньше счетчика описателей. Разница между этими двумя значениями — число прямых ссылок на структуру объекта из режима ядра, а не косвенных (через описатель);
- квота — объем пулов подкачиваемой и неподкачиваемой памяти, который добавляется в квоту процесса при создании объекта.

В нижней части вкладки **Details (Подробно)** показана информация, специфичная для объекта, если таковая доступна. Например, для символьной ссылки показывается время создания и путь к целевому объекту, а для объекта-события показывается тип и указывается, свободен ли он.

На вкладке **Security (Безопасность)** диалогового окна свойств показаны общие разрешения на доступ к объекту. Однако заметьте, что открыть можно объекты не всех типов, а разрешения определенного объекта могут запрещать просмотр его свойств.

Особый интерес представляют следующие каталоги:

- **\BaseNamedObjects** — содержит объекты, создаваемые в глобальном пространстве имен, такие как события и семафоры, а также объекты, которые создаются в локальном пространстве имен процессами, работающими в нулевом сеансе служб терминалов.
- **\Sessions*n*** — содержит закрытые данные сеансов быстрого переключения пользователей и служб терминалов, указанных номерами *n*, где *n* равно или больше 1.
- **\Sessions*n*\BaseNamedObjects** — содержит объекты, такие как события и семафоры, созданные в локальном пространстве имен процессов, работающих в сеансах быстрого переключения пользователей и служб терминалов, указанных номерами *n*.
- **\Sessions\0\DosDevices*LUID*** — содержит закрытые данные сеансов LSA, указанного в имени каталога локально уникальным идентификатором (LUID), включая подключения SMB, буквы сетевых дисков и сопоставления SUBST.
- **\GLOBAL??** — содержит символьные ссылки, связывающие с устройствами глобальные имена, включая глобально определенные буквы дисков и другие имена устройств MS-DOS, такие как AUX и NUL.

\KnownDLLs и **\KnownDlls32** — имена и пути разделов для библиотек DLL, проецируемых системой при загрузке. Каталог \KnownDlls32 существует только в 64-разрядных версиях Windows и содержит список 32-разрядных версий стандартных DLL.

LoadOrder

LoadOrder (Loadord.exe) — простая утилита, показывающая примерный порядок, в котором Windows загружает драйверы устройств и запускает службы. LoadOrder работает во всех версиях Windows и не требует административных прав.

LoadOrder определяет порядок драйверов и служб на основе стартового значения, имени группы, идентификатора тега и зависимостей. Утилита показывает все эти атрибуты, кроме зависимостей (рис. 14-11). Первыми загружаются драйверы, которые запускаются при загрузке, затем драйверы, запускаемые системой, а затеем драйверы и службы с автоматической загрузкой. Заметьте, что драйверы и службы, запускаемые по запросу (вручную), не отображаются. На этапе запуска Windows загружает драйверы по группам, а в пределах группы — по идентификатору-метке. Группы загружаются в том порядке, в каком перечислены в разделе HKLM\System\CurrentControlSet\Control\ServiceGroupOrder, а метки назначаются в порядке, указанном для соответствующей группы в разделе HKLM\System\CurrentControlSet\Control\ GroupOrderList. Группы и метки, не указанные в этих разделах, игнорируются при определении порядка загрузки; LoadOrder помечает их звездочкой. В дополнение к стартовому значению, имени группы и метке, LoadOrder показывает для каждого драйвера и службы внутреннее и отображаемое имена, а также путь к образу.

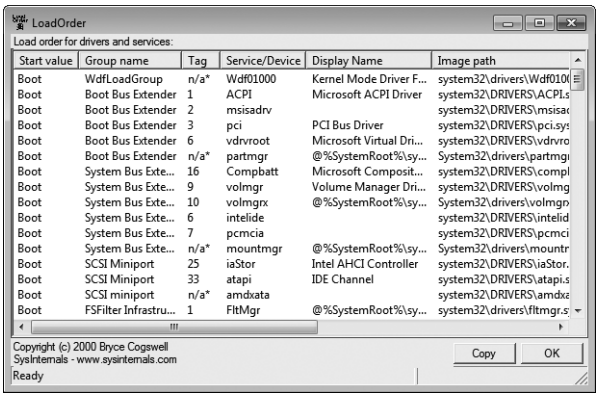


Рис. 14-11. LoadOrder

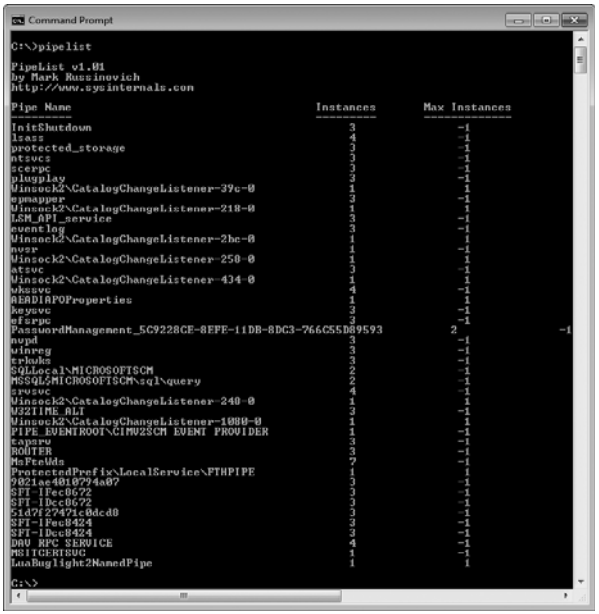


Рис. 14-12. PipeList

Чтобы скопировать данные из окна LoadOrder в буфер обмена в виде текста с разделителями-табуляторами, щелкните кнопку **Copy (Копировать)**.

Некоторые драйверы и службы могут загружаться не в том порядке, какой показывает LoadOrder. Драйверы Plug-and-Play обычно регистрируются как запускаемые по запросу, и поэтому не показываются, но они загружаются во время определения и перечисления устройств. Кроме того, LoadOrder не делает различий между службами с автоматической загрузкой («Automatic») и автоматической загрузкой с задержкой («Automatic (Delayed Start)»). Последние загружаются после служб с автоматической загрузкой.

Подробнее о том, как Windows загружает и запускает драйверы и службы, см. в пятой редакции книги «Windows Internals».

PipeList

Именованные каналы реализованы в Windows через драйвер файловой системы NPFS.sys, имя которого расшифровывается как Named Pipe File System (файловая система именованных каналов). PipeList — консольная утилита, показывающая все именованные каналы на локальном компьютере, выполняя просмотр списка каталогов этой файловой системы. Как показано на рис. 14-12, PipeList также показывает число экземпляров имен и максимальное разрешенное число экземпляров. Значение **Max Instances (Макс. экземпляров)**, равное -1, означает, что число экземпляров не ограничено.

PipeList работает во всех версиях Windows и не требует администраторских прав.

ClockRes

ClockRes — простая утилита командной строки, показывающая текущее разрешение системных часов, а также минимальный и максимальный интервалы между тактами часов. Она не требует администраторских прав.

Текущий квант времени обычно бывает больше максимального, когда какой-либо процесс, например, хост-процесс мультимедийного приложения, повышает разрешение для передачи звука и видео. В Windows 7 HTML-список изменивших разрешение таймера процессов можно получить при помощи стандартной утилиты Powercfg.exe, запущенной с командной /energy.

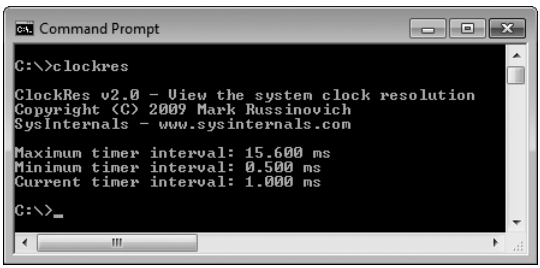


Рис. 14-13. ClockRes

Глава 15

Другие утилиты

В этой главе описаны утилиты, не имеющие отношения к диагностике и устранению неполадок. Это простые утилиты, написанные мной для собственных нужд или просто для развлечения, которые я затем опубликовал на сайте Sysinternals.

- **RegJump** запускает RegEdit и переходит по указанному пути;
- **Hex2Dec** преобразует числа из шестнадцатеричной системы счисления в десятичную и наоборот;
- **RegDelNull** ищет и удаляет разделы реестра с Null-символами в именах;
- **Хранитель экрана Bluescreen** реалистично имитирует «синий экран смерти»;
- **Ctrl2Cap** — драйвер-фильтр клавиатуры, преобразующий нажатие Caps Lock в нажатие Ctrl. Предназначен для тех, кто использует клавиатуры, на которых клавиша Ctrl расположена слева от клавиши A.

RegJump

RegJump — утилита командной строки, принимающая путь к разделу реестра, открывающая редактор реестра RegEdit и переходящая в этом редакторе по указанному пути. Корневой раздел можно указать в стандартном или сокращенном виде и даже в формате спецификатора диска PowerShell. Необязательно заключать в кавычки пути, содержащие пробелы. Все следующие команды равнозначны:

```
regjump HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control

regjump HKLM\SYSTEM\CurrentControlSet\Control

regjump HKLM:\SYSTEM\CurrentControlSet\Control
```

RegJump программно отправляет нажатия клавиш редактору RegEdit. Это значит, что в Windows Vista и последующих версиях RegJump необходимо запускать с уровнем целостности, не меньшим, чем у RegEdit. Если вы входите в группу администраторов, RegEdit требует повышения прав, так что RegJump тоже необходимо запускать с повышенными правами. Если вы

вошли как обычный пользователь, ни RegJump, ни RegEdit не требуют повышения привилегий.

Hex2Dec

Если вы проводите много времени за работой с командной строкой или консолью Windows PowerShell, утилита Hex2Dec будет удобным инструментом для преобразования чисел из шестнадцатеричной в десятичную систему счисления и наоборот, не открывая Калькулятор Windows. Просто введите в командную строку число, которое хотите преобразовать, используя префикс `x` или `0x`, чтобы указать шестнадцатеричное число. Hex2Dec интерпретирует входные данные как 64-разрядные целые числа (qword), обрабатывая десятичные значения как 64-разрядные целые числа со знаком. На рис. 15-1 показаны примеры.

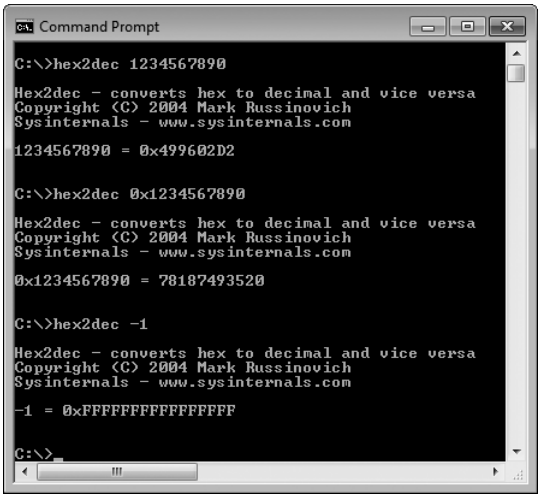


Рис. 15-1. Hex2Dec

RegDelNull

Из-за того, как собственные API Windows и ядро Windows обрабатывают строковые значения, допустимо создание и обращение к разделам и значениям реестра с null-символами в именах. Поскольку в Win32 API null-символ используется для указания конца строкового значения, невозможно обращаться или удалять такие разделы и значения при помощи интерфейсов Win32 API или стандартных редакторов реестра, использующих эти API, таких как RegEdit.

RegDelNull ищет разделы реестра, имена которых содержат null-символы, и позволяет удалять их. Укажите раздел, в котором будет выполнен поиск, и добавьте параметр -s для поиска в подразделах. Если такой раздел найден, путь к нему отображается со звездочкой вместо null-символа, и предлагает-

ся удалить этот раздел (рис. 15-2). Заметьте, что удаление разделов реестра может привести к выходу из строя приложений, использующих их.

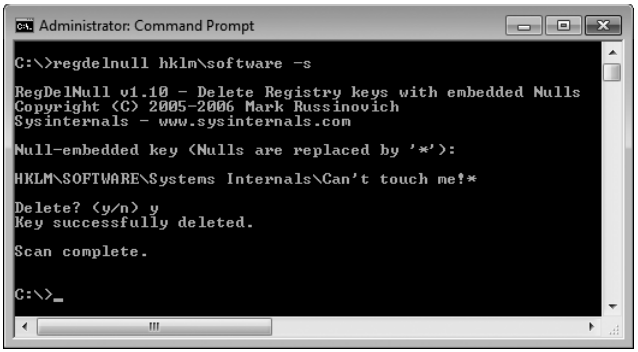


Рис. 15-2. RegDelNull

Экранная заставка Bluescreen

Эта заставка написана просто для развлечения. Она реалистично имитирует бесконечную череду крахов системы с «синим экраном смерти» (Blue Screen of Death, BSOD) и перезагрузок системы. Для каждого «сбоя» заставка случайно выбирает код ошибки и отображает реалистичные данные, соответствующие этому коду. Для имитации перезагрузки отображается экран загрузки Windows XP с индикатором хода, после которого снова происходит «сбой» (заставка уже устарела и не показывает экраны загрузки Windows Vista и Windows 7).

Чтобы установить эту заставку, скопируйте файл SysinternalsBluescreen.scr в папку System32 и выберите его в окне настройки экранной заставки. Можно также скопировать его в любую папку, щелкнуть на нем правой кнопкой мыши в Проводнике и выбрать в контекстном меню команду **Установить (Install)**. Хранитель экрана Bluescreen не входит в пакет Sysinternals, но его можно скачать отдельно с сайта Sysinternals.

Примечание В окне настройки Bluescreen имеется флажок **Fake Disk Activity (Имитировать активность диска)**, который ничего не делает в ОС выше Windows NT 4.0, после которой «Синий экран смерти» был значительно переработан.

Хранитель экрана Bluescreen работает во всех версиях Windows и не требует административных прав. Но так как он требует изменения режима отображения, он не будет работать в сеансах удаленного рабочего стола; а поскольку он использует DirectX, то может и не работать в виртуальных машинах.

Используйте этот хранитель экрана с осторожностью! Мы слышали истории о том, как пользователи много раз по-настоящему отключали питание компьютера для перезагрузки системы после фальшивого сбоя. Также мы слышали, что у одного человека, у которого эта заставка включилась во время презентации. Он равнодушно нажал клавишу и продолжил демонстра-

цию, а слушатели до того изумились, что просто перестали обращать внимание на докладчика и побежали звонить руководству с сообщением о «новом способе быстрого восстановления после 'синего экрана', на котором можно заработать состояние»...

Ctrl2Cap

Прежде чем начать работать с Windows, я проводил все свое время за UNIX-системами, у которых клавиша Control находилась там, где у стандартных клавиатур PC находится Caps Lock. Чтобы не изменять привычку, я решил изучить устройство Windows и написать драйвер режима ядра, преобразующий нажатия Caps Lock в нажатия Control. Ctrl2Cap — первая написанная мной утилита Sysinternals. Я до сих пор использую ее, и никогда не жалел об отсутствии Caps Lock.

Ctrl2Cap работает во всех версиях Windows x86 и x64. Для установки и удаления этой утилиты требуются административные привилегии.

Чтобы установить Ctrl2Cap, выполните команду ctrl2cap /install из папки, в которую распаковали файлы Ctrl2Cap. Для удаления утилиты выполните команду ctrl2cap /uninstall. В отличие от других утилит Sysinternals, упакованных в один исполняемый файл, способный самостоятельно извлекать все необходимые дополнительные файлы, архив Ctrl2Cap включает файл Ctrl2Cap.exe и несколько .sys-файлов. Во время установки Ctrl2Cap.exe определяет, какой из .sys-файлов подходит для текущей системы, копирует его в папку System32\Drivers под именем Ctrl2Cap.sys и регистрирует его как драйвер-фильтр клавиатуры.

Часть III

Поиск и устранение сбоев: загадочные случаи

Глава 16

Сообщения об ошибках

В этой главе я расскажу о способах поиска и устранения неисправностей с помощью утилит Sysinternals в тех случаях, когда основным симптомом является сообщение об ошибке. Во всех случаях, описанных в данной главе, кроме первых двух, инструментом поиска и устранения сбоев будет утилита Procmon:

- **Заблокирована папка** — типичный случай, когда выручает Procexp.
- **Невозможность обновления антивирусной программы.** В данном случае используется функция анализа системы в автономном режиме (Analyze Offline System) утилиты Autorun для восстановления работы компьютера, который не загружается.
- **Сбой при резервном копировании Lotus Notes.** Этот случай представляет для меня большой интерес. Многим читателям будет полезно узнать об этом, чтобы научиться искать потерянные DLL-файлы в Procmon.
- **Неисправность функции Play To и крах утилиты Proksi.** В этом разделе будут рассмотрены различные ошибки, включая запрет доступа с сообщением ACCESS DENIED.
- **Сбой при установке.** Как оказалось, такая ситуация возникает из-за пренебрежения правилами безопасности.
- **Отсутствие сопоставления с папкой.** Здесь сравнивается трассировка Procmon в системе с неисправностями с трассировкой в исправной системе.
- **Проблема с временными профилями реестра** представляет особый интерес, поскольку с ней сталкивались многие пользователи, а также потому, что для ее устранения используется одна из наименее известных функций Procmon — ведение журнала загрузки.

Заблокированная папка

Работая над Главой 17, я решил переименовать папку с файлами. При этом я столкнулся с неожиданной ошибкой (см. рис. 16-1): у другой программы был открытый описатель папки или вложенного в нее объекта. Убедившись в том, что я ничего не открывал из этой папки, я щелкнул **Try Again** (Попробовать снова), однако папка по-прежнему использовалась другой программой, и переименовать ее не удалось.

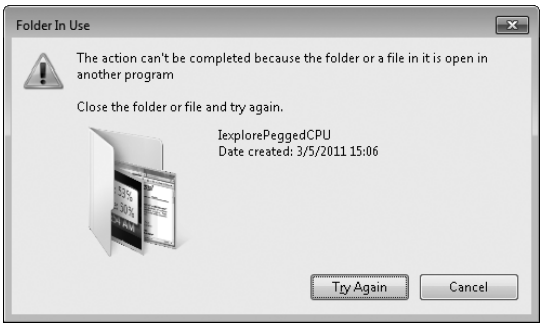


Рис. 16-1. Папка или вложенный в нее элемент открыт другой программой

Я нажал Ctrl+F в Procexp, чтобы открыть окно **Search**, ввел имя папки и щелкнул **Search**. Программа Procexp определила, что программой с открытым описателем является Microsoft Outlook (см. рис. 16-2.).

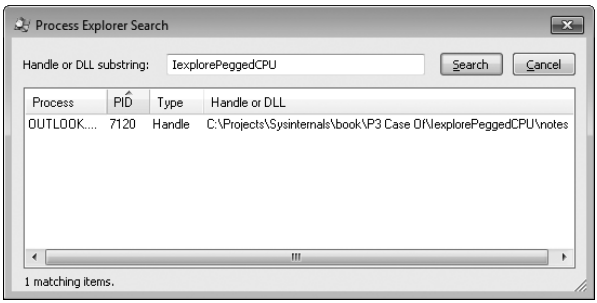


Рис. 16-2. Поиск процессов с открытыми описателями к папке IexplorePeggedCPU

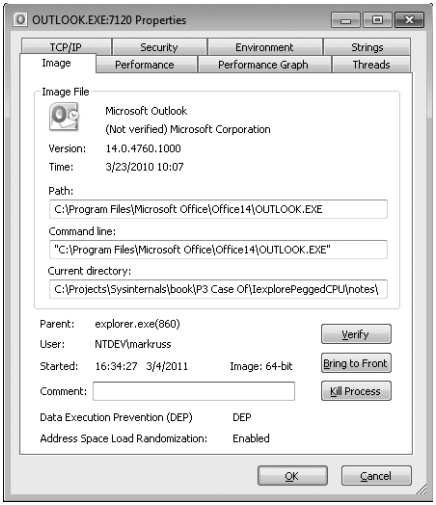


Рис. 16-3. Текущий каталог Outlook внутри папки, которую не удается переименовать

Затем я вспомнил, что я сохранил вложение из электронного письма внутри папки, которую я пытался переименовать. Я открыл окно **Properties**

процесса Outlook.exe и увидел на вкладке **Image**, что программа еще использует ту самую папку (см. рис. 16-3). Я мог бы решить проблему, закрыв Outlook, однако вместо этого я просто сохранил вложение из случайного письма в другую папку, таким образом она стала текущим каталогом Outlook, освободив описатель папки, которую я не мог переименовать. Так решилась эта проблема.

Невозможность обновления антивируса

После удаления вируса, описанного в главе 18, Пол (друг Аарона) порекомендовал своему сыну регулярно обновлять антивирус, и решил показать пример, сделав это на своем компьютере. К сожалению, это привело лишь к тому, что компьютер перестал загружаться.

Да, программы следует регулярно обновлять, поэтому Пол обновил бесплатную антивирусную программу на своем компьютере под управлением ОС Windows XP. В процессе перезагрузки на дисплее компьютера появился индикатор загрузки Windows XP, за которым последовал синий экран. Последующие перезагрузки привели к тому же результату.

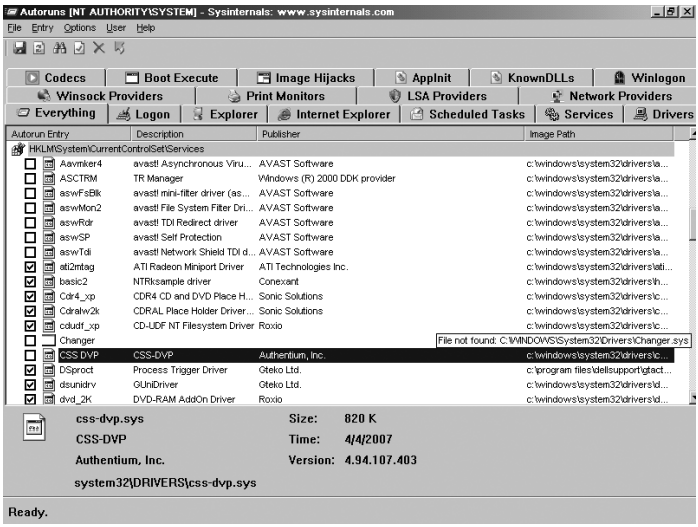


Рис. 16-4. Autoruns анализирует систему в автономном режиме, отключая неисправные антивирусные приложения и другие ненужные элементы

Пол, разумеется, пригласил к себе Аарона, и тот пришел, облаченный в поношенную футболку с надписью «Нет, я не починю ваш компьютер». Аарон, пожалуй, мог бы решить проблему в безопасном режиме или с помощью восстановления системы, но это было бы слишком просто для него (он хотел убедиться в том, что дело в неисправном антивирусе). Вместо этого он загрузился со старого компакт-диска со средой WinPE. После этого он запустил программу Autoruns, выбрал **File | Analyze Offline System**, указав папку C:\Windows на жестком диске, а также один из профилей в папке C:\Documents and Settings.

Старый экземпляр WinPE не смог проверить подлинность подписей, поэтому Аарон решил скрыть элементы Microsoft и Windows, пропустив проверку подписей, в надежде на то, что ни один из модулей не будет выдавать себя за продукт Microsoft. Помимо неисправных точек ASEP антивируса, программа Autoruns обнаружила ряд других ненужных и устаревших служб и драйверов. Аарон отключил их все, как показано на рис.16-4, и перезагрузил компьютер.

Компьютер перезагрузился без проблем. Войдя в систему, Пол засомневался, стоит ли рискнуть и снова попытаться обновить антивирус. Посоветовавшись с Аароном, он перешел на новую антивирусную программу (рис. 16-5), удалив старый антивирус. Так решилась эта проблема.

Рис. 16-5. Microsoft Security Essentials: бесплатная и надежная антивирусная защита

Сбой при резервном копировании Lotus Notes

Один из системных администраторов сообщил, что резервные копии Lotus Notes перестали восстанавливаться с сообщением об ошибке, показанным на рис. 16-6.

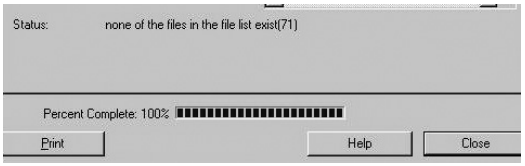


Рис. 16-6. Неисправность резервных копий Lotus Notes

Он проверил журнал приложения резервного копирования, в котором появилось сообщение о неудаче чтения DLL-файла nnotes.dll (рис. 16-7). Ему показалось странным, что путь к указанному DLL-файлу находился в папке, принадлежавшей совсем другому приложению.

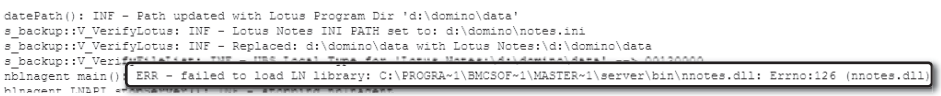


Рис. 16-7. Сообщение о неудаче при чтении nnotes.dll в журнале приложения резервного копирования

Системный администратор запустил Procmon и попытался начать резервное копирование. Он назначил фильтр по имени приложения резервного копирования и его дочерних процессов, он нажал Ctrl+F, чтобы найти строку «\nnotes.dll» в трассировке. Нашлось несколько вызовов QueryOpen (рис. 16-8), пытающихся открыть файл в различных папках, каждый раз неудачно. При этом отображалась надпись NAME NOT FOUND. Наконец, файл был найден в C:\Program Files\BMC Software\MasterCell\server\bin (по словам системного администратора, некоторые из этих папок находились в переменной окружения PATH, а остальные были добавлены приложением резервного копирования в период выполнения).

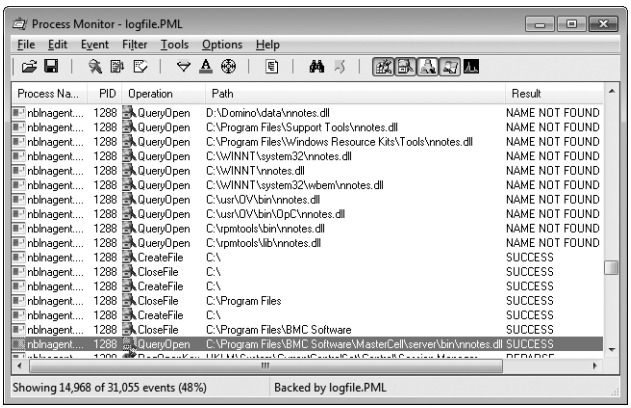


Рис. 16-8. Поиск файла nnotes.dll

Вскоре после открытия той копии nnotes.dll процесс спроецировал DLL-файл в свое виртуальное адресное пространство. Это действие сопровождается событием *Loadlmage* (см. первую строку на рис. 16-9). Почему же не удалось загрузить DLL-файл? Как видно из рис. 16-9, вскоре после загрузки nnotes.dll процесс начал поиск nxmlproc.dll, просматривая все пути из переменной PATH по порядку, однако искомый файл найден не был.

Системный администратор запустил утилиту Dumpbin из Microsoft Visual Studio, для просмотра зависимостей DLL и понял, что nnotes.dll зависит от nxmlproc.dll. Поэтому сразу после загрузки nnotes.dll начинался поиск nxmlproc.dll, а отсутствие nxmlproc.dll мешало загрузить nnotes.dll.

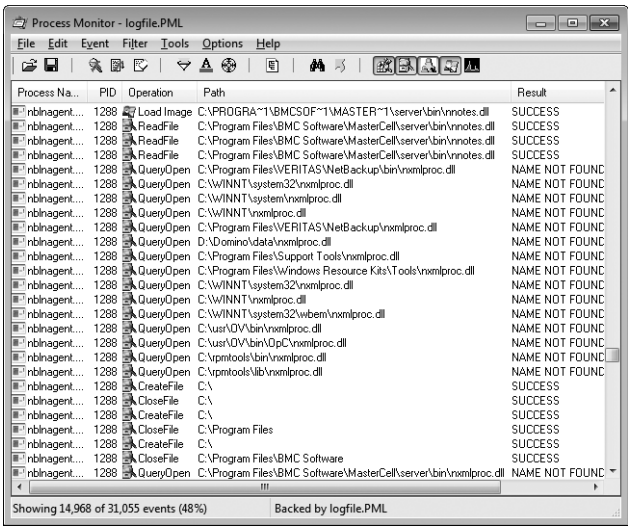


Рис. 16-9. Приложение резервного копирования успешно загружает nnotes.dll, но не может найти nxmlproc.dll

После nxmlproc.dll искали на дисках C и D, в конце концов этот файл обнаружился в папке D:\Domino (рис. 16-10), которую при поиске не просматривали. Что интересно, копия nnotes.dll лежала в той же папке (рис. 16-11). Администратор вставил «D:\Domino» в переменную PATH сразу после стандартных путей Windows по умолчанию, перезагрузил компьютер и резервное копирование заработало как часы. Так была решена эта проблема.

```
C:\>dir /s nxmlproc.dll
 Volume in drive C is SYSTEM
 Volume Serial Number is 6CE9-0DF1
File Not Found

C:\>D:

D:\>dir /s nxmlproc.dll
 Volume in drive D is Data
 Volume Serial Number is A4F8-810F

 Directory of D:\Domino

27.09.2006  11:42         3.215.360 nxmlproc.dll
               1 File(s)      3.215.360 bytes

     Total Files Listed:
               1 File(s)      3.215.360 bytes
               0 Dir(s)  251.235.532.800 bytes free
```

Рис. 16-10. Поиск nxmlproc.dll на дисках C и D

```
D:\>dir /s nnotes.dll
 Volume in drive D is Data
 Volume Serial Number is A4F8-810F

 Directory of D:\Domino

15.05.2007  11:40        14.282.752 nnotes.dll
               1 File(s)     14.282.752 bytes

     Total Files Listed:
               1 File(s)     14.282.752 bytes
               0 Dir(s)  251.235.536.896 bytes free
```

Рис. 16-11. Поиск nnotes.dll

Сбой воспроизведения в медиаплеере

Один из пользователей пытался использовать функцию «Воспроизвести на» (Play To) в Windows 7 для проигрывания песни в медиаплеере и получил непонятное сообщение об ошибке (рис.16-12). Тем не менее, другие песни из медиабиблиотеки этого пользователя проигрывались без проблем.

Рис. 16-12. Ошибка с сообщением «Error occurred on your device»

После этого пользователь вновь столкнулся с ошибкой, на этот раз в процессе мониторинга активности системы с помощью программы Procmon. После установки фильтра по файлу песни трассировка показала успешные операции с Wmplayer.exe и единственный результат с «ACCESS DENIED» для программы Wmpnetwk.exe. (рис. 16-13).

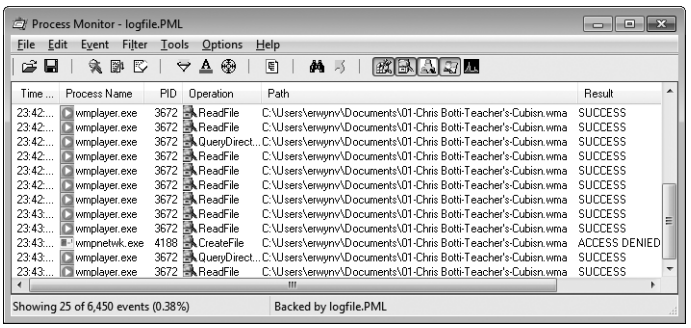


Рис. 16-13. Успешные операции Wmplayer.exe и неудачная операция Wmpnetk.exe

Он также заметил, что песни, которые успешно проигрывались, находились в папке по умолчанию **Music**, а песня, которую не удалось воспроизвести, — в папке **Documents**. Сравнив их разрешения, он обнаружил, что песни, которые успешно воспроизводились, давали разрешение на чтение и выполнение службе WMPNetworkSvc[1]. Он добавил это разрешение для файла, который не удавалось воспроизвести (рис.16-14), и песня стала звучать без сбоев. Проблема была решена.

---

[1] В Windows Vista и выше службам присвоены идентификаторы защиты (SID), благодаря чему появилась возможность предоставлять или запрещать доступ отдельным службам.

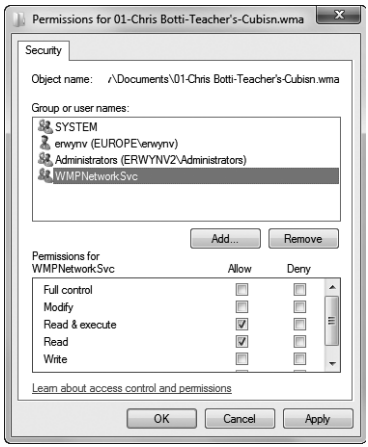


Рис. 16-14. Предоставление разрешения на доступ службе WMPNetworkSvc

Крах утилиты Proksi

Один из пользователей работал с утилитой Proksi более года, а потом утилита стала завершаться с крахом. Для диагностики он запустил Procmon, а затем — проблемную утилиту. После краха трассировку прекратили. Просканировав результаты (рис. 16-15), он обнаружил результат «ACCESS DENIED» при попытке открыть файл со стандартными разрешениями на запись.

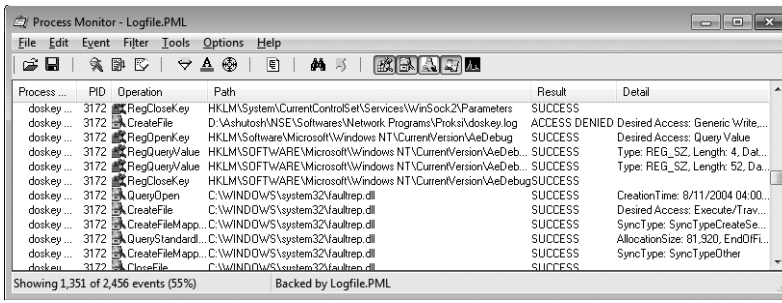


Рис. 16-15. Procmon выводит сообщение «ACCESS DENIED», затем отладчик обрабатывает крах

Пользователь открыл вкладку **Security** в окне свойств файла в Проводнике, однако не обнаружил никаких ошибок. Затем он заметил, что на вкладке **General** установлен флажок **Read-Only** (рис.16-16). Он снял его, и программа стала работать без сбоев.

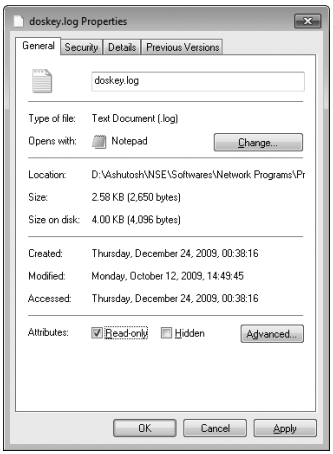


Рис. 16-16. Отказ в доступе в результате установки флажка Read-Only

Сбой при установке

Один из клиентов моего соавтора работал со сканерами «Kodak», к которым прилагались компакт-диски с необходимым программным обеспечением. Когда администратор вставил компакт-диск в дисковод, автозапуск Windows Vista сработал некорректно: открылось окно автозапуска, однако не появились параметры установки ПО. Администратор открыл папку в Проводнике и запустил файл autorun.exe, чтобы начать установку вручную. Согласившись на повышении привилегий UAC, администратор увидел сообщение о странной ошибке, как будто программа установки не смогла найти нужную версию ОС (рис. 16-17).

Рис. 16-17. Сообщение об ошибке при установке приложения

Поиск и устранение сбоя

Аарон понял, что автор программы установки считал, что Windows XP настолько совершенна, что Microsoft не будет выпускать других версий, поэтому предусматривать их наличие при проверке не стоит. Он включил режим совместимости с Windows XP, который, ко всему прочему, сообщает программе ложную информацию об операционной системе, и попробовал установить программу снова с тем же результатом. Кстати, установка прекрасно выполнялась в свежих копиях Windows Vista, к которым не применялась корпоративная политика.

Он запустил Procmon, снова вызвал программу установки и в момент появления сообщения об ошибке остановил трассировку Procmon. Аарон отфильтровал, перетащив значок в виде перекрестия с панели управления Procmon на сообщение об ошибке, только события, связанные с процессом-владельцом окна Setup.exe (рис. 16-18).

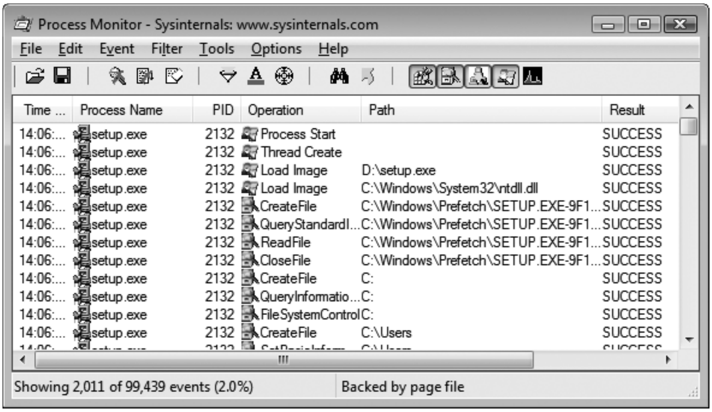


Рис. 16-18. Procmon после установки фильтра «Include Process From Window»

Поскольку в заголовке сообщения об ошибке присутствовала цифра «0», Аарон подумал, что проблема возникает из-за того, что программа не может что-то найти, поэтому он щелкнул правой кнопкой столбец **Result** и исключил события, не представляющие интереса: SUCCESS, FAST I0 DISALLOWED, FILE LOCKED WITH ONLY READERS, REPARSE, BUFFER OVERFLOW, и END OF FILE (обычно Аарон исключает стандартные результаты, а не включает потенциально тревожные, иначе легко выкинуть нужную информацию).

Когда он посмотрел на оставшиеся записи, то увидел один явно выделяющийся элемент — «DoesNotExist» среди путей в конце перечня результатов. Он применил функцию подсветки Procmon, чтобы выделить их в контексте окружающих событий (см. рис. 16-19).

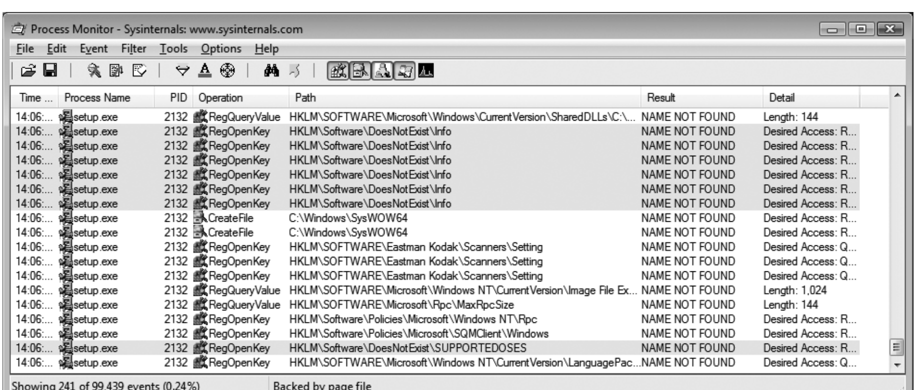


Рис. 16-19. Подсветка строки «DoesNotExist» в отфильтрованных результатах

Поскольку из контекста было неясно, что произошло непосредственно перед неудачным поиском, он удалил из фильтра Procmon правило, исключающее результаты SUCCESS. Как видно из рис. 16-20, перед обращением к разделу реестра HKLM\Software\DoesNotExist\Info выполнялось множество обращений к D:\setup.ini, а затем несколько к D:\autorun.inf.

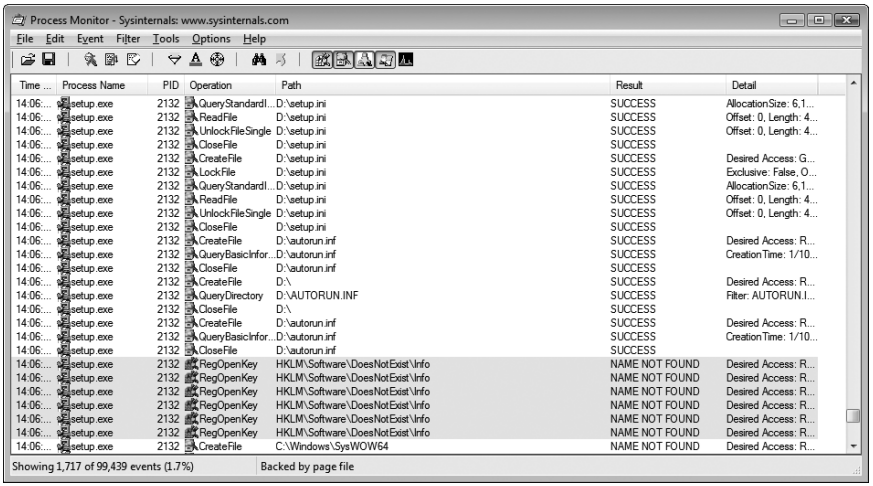


Рис. 16-20. Вывод результатов типа SUCCESS перед неудачным обращением к реестру

Аарон открыл меню **Properties** первого события *RegOpenKey* и посмотрел на стек вызовов (рис. 16-21), чтобы понять, как и зачем процесс Setup.exe пытался открыть этот раздел. Строка 12 стека показывала, что компонент программы установки вызывал *GetPrivateProfileStringA*, что и привело (в строке 7) к попытке открыть этот раздел реестра.

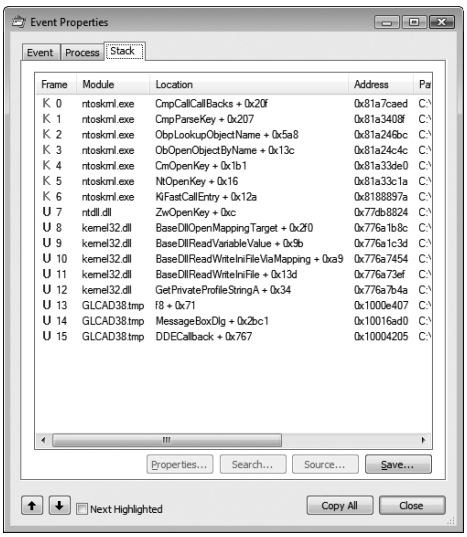


Рис. 16-21. Стек вызовов неудачной попытки открытия HKLM\Software\DoesNotExist\Info

GetPrivateProfileString — это API-функция, которой Windows-разработчики могут пользоваться для чтения старых INI-файлов 16-разрядных версий Windows. Согласно документации, эти вызовы перенаправляются в реестр с помощью раздела IniFileMapping. Аарон обнаружил в IniFileMapping строку, направляющую запрос autorun.inf в «DoesNotExist» (рис. 16-22), удалил его и перезагрузил компьютер, после чего установка прошла успешно.

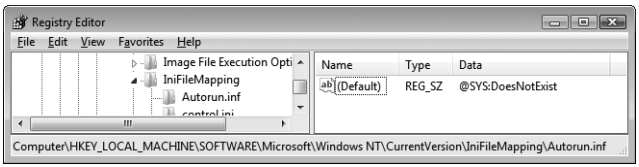


Рис. 16-22. Элемент IniFileMapping перенаправляет запрос Autorun.inf в несуществующий раздел реестра

Анализ проблемы

Аарон обнаружил конкретную причину неудачной установки, однако ему нужна была ее первопричина и объяснение неверной конфигурации IniFileMapping.

IniFileMapping

Раздел IniFileMapping появился в Windows NT 3.1. Когда программы используют API-интерфейсы ini-файлов для обращения к файлам, IniFileMapping может перенаправлять обращения к кустам MACHINE или USER (HKLM или HKCU). Раздел IniFileMapping разработан для поддержания совместимости с устаревшими приложениями, использовавшим INI-файлы вместо реестра, что препятствовало масштабированию и не давало возможности разным пользователям иметь собственные настройки.

Autorun.inf

При вставке съемного носителя, такого как компакт-диск или флеш-накопитель USB Windows обнаруживает новый диск и Проводник проверяет наличие файла Autorun.inf в корневой папке носителя. Autorun.inf — это текстовый файл в формате INI-файла (то есть, с именами разделов в квадратных скобках, содержащих пары «имя=параметр»). Файл может содержать записи, сообщающие Проводнику, какой значок следует назначить диску и какое действие по умолчанию следует предлагать пользователю, а в некоторых случаях просто запускает нужную программу. Этот механизм позволяет запускать программу установки автоматически после вставки компакт-диска. Существуют настройки реестра и групповой политики, управляющие автозапуском и автоматическим воспроизведением (см. статью 967715 базы знаний Microsoft Knowledge Base по адресу http://support.microsoft.com/kb/967715).

Проблема с автозапуском заключается в том, что по умолчанию он действует и для перезаписываемых дисков, включая флеш-накопители. Черви,

такие как Conficker, могли распространяться через такие устройства, записывая на диск Autorun.inf и свою копию. После этого вредоносное ПО могло заражать другие компьютеры после подключения к ним диска. Все это усугублялось ошибкой в применении настроек, которые, как предполагалось, должны были отключать автоматическое воспроизведение. Позднее эта ошибка была исправлена. Более того, в новых версиях Windows автоматическое воспроизведение для перезаписываемых устройств теперь отключено по умолчанию. Подробнее об этом написано в статье 971029 Knowledge Base (*http://support.microsoft.com/kb/971029*). Функции автозапуска и автоматического воспроизведения все еще работают на CD- и DVD-дисках, поскольку распространение червей через эти устройства менее вероятно и пока удобство перевешивает риск.

Почему в разделе IniFileMapping появился параметр для Autorun.inf?

Пару лет назад один блоггер описывал хитрый трюк для отключения автоматического воспроизведения на всех носителях. Трюк был возможен благодаря тому, что Autorun.inf имеет формат ini-файла, а Проводник использует API-функции ini-файлов для его чтения. За счет создания для Autorun.inf параметра IniFileMapping, перенаправляющего обращение к несуществующему разделу реестра, записи автоматического воспроизведения становятся недоступными. Автор утверждал, что при этом имеется только один недостаток: пользователи сами должны искать исполняемые файлы. Когда вредоносные программы все чаще стали распространяться через перезаписываемые устройства, группа реагирования на компьютерные происшествия (CERT) Университета Карнеги-Меллона и другие организации, занимающиеся вопросами информационной безопасности, стали рекомендовать этот трюк, утверждая, что он не вызывает побочных эффектов. С тех пор данный параметр в обязательном порядке вносится в реестр во многих организациях.

Почему не удалось установить это приложение?

Оказалось, что файл Autorun.inf на установочном компакт-диске «Kodak» содержал отнюдь не только записи для автоматического воспроизведения:

```
[autorun]
open=autorun.exe

[Info]
Dialog=Kodak i610/i620/i640/i660 Scanner
Model=600
ModelDir=kds_i600
Prog ramG roup=i 610,i 620,i 640,i 660

[Versions]
CD=04040000
FIRMWARE=04000300
ISISDRIVER=2.0.10711.12001
```

```
ISISTOOLKIT=57.0.260.2124
KDSMM=01090000
PKG=02010000
SVT=06100000
TWAIN=09250500

[Install]

[SUPPORTEDOSES]
WIN=WINVISTA WINXP WIN2K

[REQUIREDSPS]
WINXP=1
WIN2K=3
```

«Kodak» и другие производители используют Autorun.inf не только для автоматического воспроизведения, но и в качестве ini-файла общего назначения для своих программ установки. Разумеется, программа установки использует для чтения файла стандартные API-интерфейсы, однако IniFileMapping перенаправляет их в несуществующий раздел реестра, в результате установка не удается. Надо сказать, что «Kodak» делает это совершенно законно: нигде не запрещено помещать в Autorun.inf иные настройки, предусмотренные приложением.

Может ли клиент решить проблему, скопировав содержимое компакт-диска на жесткий диск и запустить его оттуда? Нет. IniFileMapping применяется к любому файлу с именем «Autorun.inf», независимо от его расположения.

Мораль в том, что установка провалилась из-за уверенности в «отсутствии побочных эффектов», не подкрепленной тестированием. Поскольку новые умолчания и доступные настройки политики автоматического воспроизведения значительно снижают угрозу распространения вирусов через USB-накопители, примочки вроде IniFileMapping могут и не сработать. Аарон посоветовал клиенту удалить этот раздел из реестра и положиться на защитные меры, принятые в новых системах.

Отсутствие сопоставлений с папкой

Один из пользователей обнаружил, что всякая попытка открыть любую папку в Проводнике приводит к появлению сообщения об ошибке, изображенного на рис. 16-23. Это происходило независимо от способа, которым пытались открыть папку: двойной щелчок папки на рабочем столе, через значок Компьютер, Панель управления, Документы, Изображения или другие папки меню Пуск.

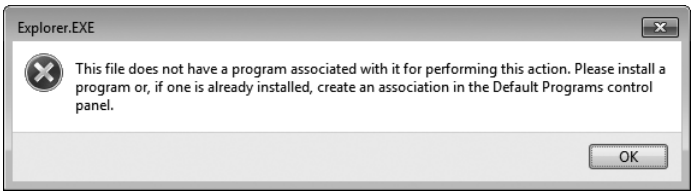


Рис. 16-23. Сообщение об ошибке при попытке открыть папку

Сопоставления программ хранятся в улье реестра HKEY_CLASSES_ROOT, поэтому пользователь предположил, что там мог быть отсутствующий или поврежденный файл. Он решил, что лучше всего будет сравнить результаты Procmon на компьютере, где возникла проблема, и на компьютере, где такой проблемы нет.

Procmon может захватывать большое количество данных за короткое время, поэтому пользователь стремился максимально сузить поле поиска. Он запустил Procmon с помощью параметра командной строки **/noconnect**, чтобы не начинать захват событий сразу, но и не пропустить ошибку. Затем он нажал Ctrl+E, чтобы запустить захват, дважды щелкнул папку и нажал Ctrl+E, чтобы остановить захват сразу после появления сообщения об ошибке. После этого он перетащил значок в виде перекрестья с панели инструментов Procmon на сообщение об ошибке, чтобы отфильтровать события данного процесса. Поскольку Explorer.exe управляет всем рабочим столом, в том числе панелью задач, областью уведомлений и т.д., пользователь решил сократить число отображаемых объектов до потока, отображающего сообщение об ошибке. Он щелкнул заголовки столбцов правой кнопкой мыши, добавил столбец **Thread ID (TID)** и поставил его рядом со столбцом **PID**. Полагая, что ему нужен поток с наибольшей активностью, он применил инструмент **Count Occurrences** (рис. 16-24) и добавил найденный поток его в фильтр. После этого он сохранил отфильтрованную трассировку командой **Save**.

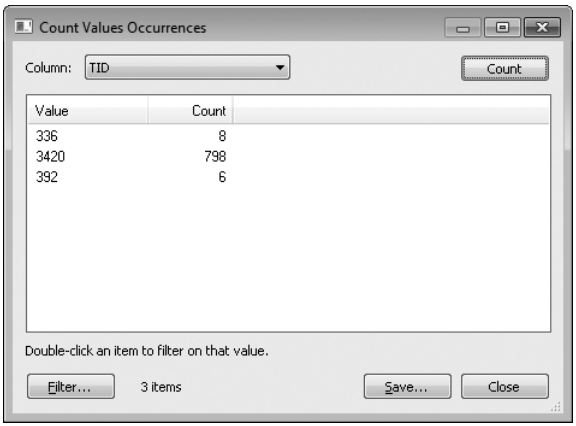


Рис. 16-24. Поиск потока с наибольшей активностью

После этого пользователь воспроизвел все действия на другом компьютере, где данная проблема отсутствовала. Поскольку сообщение об ошибке не отображалось, он остановил захват, когда открылось окно папки, отфильтровал трассировку по имени процесса Explorer.exe, которому принадлежало окно папки, и сохранил результаты в файл.

Пользователь открыл трассировки рядом, добавил в каждую столбец **TID**. Результаты исправной системы содержали гораздо больше событий. Предположив, что проблема связана с реестром, пользователь скрыл остальные классы событий с помощью кнопок панели инструментов. После этого он начал искать записи, совпадающие в «исправной» и «неисправной» трассировке. Найдя этот поток, он отфильтровал по нему «исправную» трассировку. Отыскав начало серии идентичных событий, он щелкнул правой кнопкой мыши первое событие серии в обеих трассировках и выбрал параметр **Exclude Events Before**, чтобы обе трассировки начинались с одной точки (рис. 16-25).

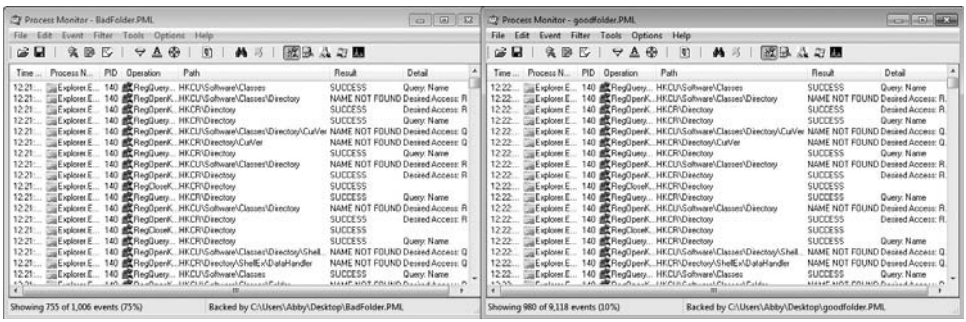


Рис. 16-25. Сравнение трассировок Procmon

Просматривая постранично результаты в поисках различий, он вскоре обнаружил операцию *RegOpenKey* в HKCR\Folder\shell\open\command, которая привела к результату NAME NOT FOUND в «неисправной» трассировке и SUCCESS в «исправной» (рис. 16-26). С помощью Regedit он экспортировал этот раздел с исправной машины и импортировал в реестр неисправной. Это несложное действие помогло решить проблему.

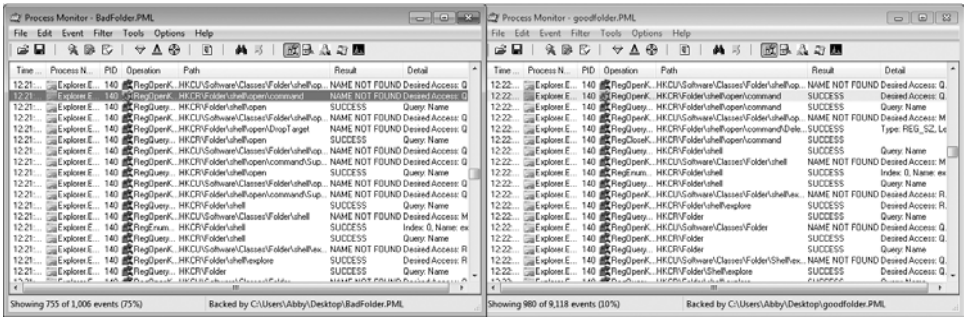


Рис. 16-26. Поиск различий между трассировками Procmon

Визуальное сравнение трассировок иногда требуется, если различий много или инструмент типа WinDiff не помогает, но в данном случае WinDiff мог бы ускорить процесс. Только нужно было бы исключить столбцы **Time of Day**, **PID** и **TID**, поскольку в трассировках они всегда различаются. Сохранив отображаемые события (исключая события профилирования) в CSV-файлы, эти файлы можно сравнить с WinDiff и отсутствующий раздел реестра сразу был бы обнаружен (рис. 16-27).

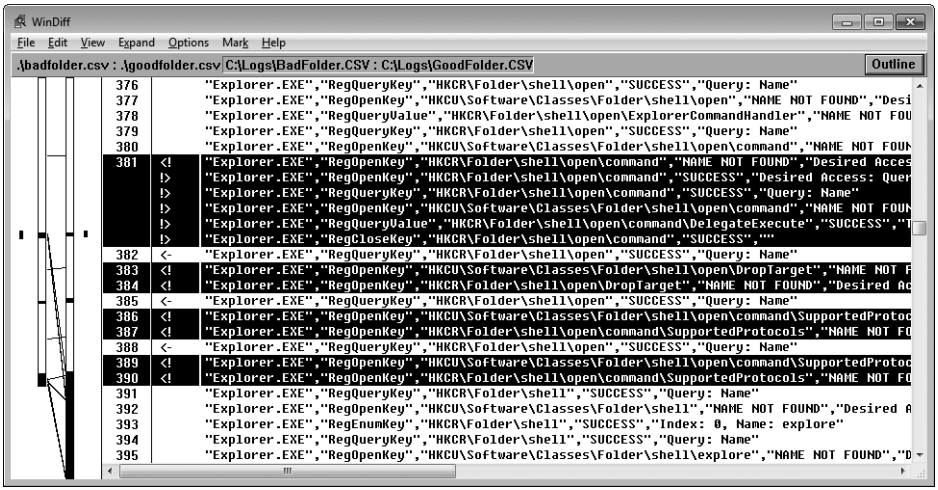


Рис. 16-27. Сравнение трассировок Procmon с помощью WinDiff

Проблема с временными профилями реестра

История данной проблемы началась с обращения в службу поддержки Microsoft из-за ошибки, периодически возникающей при входе в систему и заставляющей Windows создавать временный профиль для пользователя.

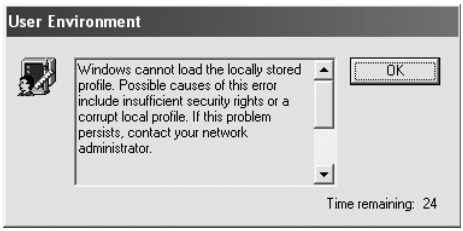


Рис. 16-28. Ошибка загрузки профиля пользователя при входе в систему

Профиль пользователя состоит из папки файловой системы %User-Profile%, в которую приложения сохраняют файлы конфигураций и данных пользователей, а также файла куста реестра, хранящегося в папке %UserProfile%\Ntuser.dat и загружаемого процессом Winlogon при входе пользователя в систему. Приложения сохраняют пользовательские настройки в кусте реестра, вызывая функции реестра, обращающиеся к корневому

разделу HKEY_CURRENT_USER (HKCU). Потеря пользователями доступа к своим профилям — всегда большая проблема, поскольку они при этом теряют все свои настройки и доступ к файлам, хранящимся в их профилях. В большинстве случаев пользователи связывались со службой поддержки компании, которая советовала им пытаться перезагружать компьютер пытаться входить в систему, пока проблема не решится сама собой.

Как обычно служба поддержки Microsoft начала с вопросов о конфигурации системы, перечне установленных программ, а также обо всех изменениях, которые были в последнее время внесены в компьютеры компании. В данном случае примечателен был тот факт, что во всех системах, где возникла проблема, был обновлен клиент ICA от Citrix Corporation (приложение удаленного рабочего стола). Представители Microsoft связались со службой поддержки Citrix, чтобы узнать о возможных проблемах с новой версией клиента, каковых не оказалось.

Не будучи уверенными, что причиной проблемы с профилями стало обновление клиента ICA, сотрудники службы поддержки Microsoft порекомендовали клиенту включить ведение журнала профиля (см. статью 221833 «How to enable user environment debug logging in retail builds of Windows» базы знаний Microsoft Knowledge Base по адресу *http://support.microsoft.com/221833*). Необходимые изменения были сделаны, но пользователю это не помогло. Тогда сделали копию журнала профиля из %SystemRoot%\Debug\UserMode\Userenv.log и отправили в службу поддержки Microsoft. В журнале не нашлось исчерпывающего ответа, но в нем был ключ к разгадке: указание о том, что профиль пользователя не загрузился из-за ошибки 32: ERROR_SHARING_VIOLATION (рис. 16-29).

Рис. 16-29. Запись в Userenv.log о неудаче загрузки профиля из-за ошибки общего доступа

Когда процесс открывает файл, он указывает, какие виды общего доступа он разрешает для файла. Например, может быть разрешено чтение, но не запись в файл. Отметка о нарушении совместного доступа в файле журнала означает, что какой-то процесс уже открыл улей реестра пользователя способом, несовместимым с тем способом, которым этот файл должен открыть процесс входа в систему.

Между тем, все больше и больше клиентов по всему миру стали обращаться в службы поддержки Microsoft и Citrix с той же проблемой. Все они установили новый клиент ICA. Служба поддержки Citrix сообщила, что ошибку, видимо, вызывает один из процессов клиента ICA, Ssonvr.exe. В ходе установки клиент ICA регистрирует DLL провайдера Pnsson.dll, которую вызывает уведомляющее приложение многосетевого доступа Windows (Windows Multiple Provider Notification Application) (%SystemRoot%\System32\Mpnotify.exe) во время загрузки системы. Mpnotify.exe запускает-

ся при входе в систему процессом Winlogon. DLL-файл уведомителя Citrix асинхронно запускает процесс Ssonvr.exe относительно входа пользователя в систему (рис. 16-30). Единственный недостаток данной гипотезы заключался в том, что, по мнению разработчиков Citrix, процесс не пытался загружать профиль реестра пользователя или даже читать какие-либо его разделы или параметры. В итоге и Microsoft и Citrix зашли в тупик.

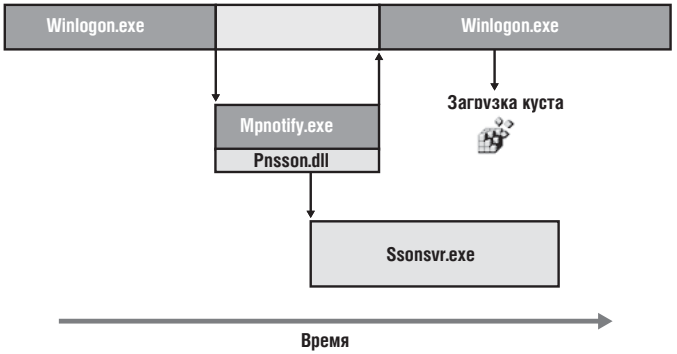


Рис. 16-30. Асинхронный запуск Ssonsvr.exe при входе пользователя в систему

В Microsoft подготовили версии Winlogon и ядра для сбора дополнительной диагностической информации, и попыталась воспроизвести проблему в тестовых системах идентичной конфигурации, но не преуспели. Не удалось воспроизвести проблему и с помощью измененных образов Windows. Скорее всего, хронометраж процессов в таких образах отличался так сильно, что проблема не возникала. Тогда инженер службы поддержки Microsoft предложил записать трассировку активности при входе в систему с помощью Procmon.

Существует два способа настройки Procmon для записи операций при входе. Первый — использование утилиты PsExec от Sysinternals для запуска Procmon в неинтерактивной оконной станции в сеансе 0[2], чтобы программа продолжала работать и после выхода из системы, а затем вновь войти в систему. Второй — включение журнала загрузки для регистрации активности с ранних ее этапов и во время входа в систему. Инженер выбрал второй способ и посоветовал клиенту запустить Process Monitor в одной из систем, регулярно сталкивавшихся с проблемой, выбрать Enable Boot Logging в меню Options и выполнить перезагрузку, повторяя все действия, пока проблема не воспроизведется. Эта процедура настраивает драйвер Process Monitor для регистрации активности процессов с самого начала загрузки в файле %SystemRoot%\Procmon.pmb. Когда клиент столкнулся с проблемой в очередной раз, ему пришлось перезапустить Process Monitor. В результате драйвер остановил ведение журнала, и программа Process Monitor предложила преобразовать журнал в стандартный формат журнала Process Monitor.

---

[2] Подробнее об оконных станциях и сеансе 0 см. в главе 2, а о запуске Procmon с помощью PsExec — в главе 4.

После нескольких попыток пользователь скопировал журнал загрузки и отправил его службе поддержки Microsoft. Проверив журнал, инженеры обнаружили, что нарушение общего доступа возникает при попытке Winlogon загрузить пользовательский куст реестра (рис. 16-31). После просмотра операций, предшествовавших ошибке, стало ясно, что именно Ssonsvr.exe был процессом, открывавшим куст реестра. Оставался вопрос: почему Ssonsvr.exe делал это?

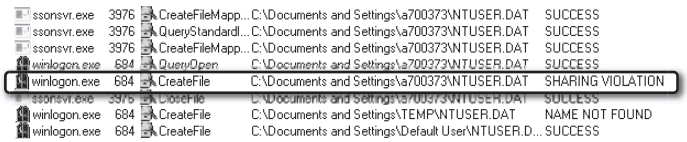


Рис. 16-31. Ssonsvr.exe открывает ntuser.dat, что вызывает нарушение общего доступа при открытии куста процессом Winlogon.exe

Для ответа на этот вопрос инженеры воспользовались трассировкой стека в Process Monitor, которая захватывает стек вызовов для каждой операции. Стек вызовов представляет собой хранилище вызовов функций, отвечающий за ту или иную операцию. Просматривая стек вызовов, зачастую удается определить первопричину операции, которую не всегда можно выявить по выполнившему ее процессу. Например, можно узнать, что операцию выполнила загруженная в процесс DLL, а если имеются символы для вызывающего образа, то в стеке можно даже прочитать имена соответствующих функций.

Просмотр стека для операции открытия файла Ntuser.dat процессом Ssonsvr.exe (рис. 16-32) показал, что в действительность операцию инициировал не Ssonsvr.exe, а Windows Logical Prefetcher.

```
8  ntkrnlpa.exe  nt!IopfCallDriver+0x31
9  ntkrnlpa.exe  nt!ObpLookupObjectName+0x53c
10 ntkrnlpa.exe  nt!ObOpenObjectByName+0xea
11 ntkrnlpa.exe  nt!IopCreateFile+0x407
12 ntkrnlpa.exe  nt!IoCreateFile+0x8e
13 ntkrnlpa.exe  nt!CcPfGetSectionObject+0x91
14 ntkrnlpa.exe  nt!CcPfPrefetchSections+0x2b7
15 ntkrnlpa.exe  nt!CcPfPrefetchScenario+0x7b
16 ntkrnlpa.exe  nt!CcPfBeginAppLaunch+0x158
17 ntkrnlpa.exe  nt!PspUserThreadStartup+0xeb
18 ntkrnlpa.exe  nt!KiThreadStartup+0x16
```

Рис. 16-32. Код Prefetcher, вызывающий IoCreateFile для загрузки Ntuser.dat

Logical Prefetcher, появившийся в Windows XP, представляет собой компонент ядра, который отслеживает процесс в течение первых 10 секунд после запуска и регистрирует каталоги и порции файлов, к которым процесс обращается, в файле %SystemRoot%\Prefetch. Чтобы различать исполняемые файлы с одинаковыми именами, но расположенные в разных папках, Logical Prefetcher дает им имена, составленные из имени файла и хеш-строки пути к нему, например, NOTEPAD.EXE-D8414F97.pf. Для просмотра файлов и папок, помеченных Logical Prefetcher для упреждающего чтения в ходе последнего запуска приложения удобна утилита Strings от Sysinternals:

```
strings prefetch-file
```

При следующем запуске приложения Logical Prefetcher, исполняемый в контексте первого потока процесса, ищет файлы для упреждающего чтения. Если таковые есть, он загружает в память метаданные соответствующих папок, если это еще не сделано. После этого Logical Prefetcher проецирует в память порции файлов, к которым процесс обращался в последний раз. Logical Prefetcher ускоряет запуск приложений, заменяя множество случайных операций файлового ввода-вывода, имеющих место при обычном запуске приложений, небольшим числом операций по загрузке крупных порций данных.

Тем не менее, причастность Logical Prefetcher к проблеме с профилями вызвала новые вопросы. Почему программа решила заранее прочесть файл куста пользователя в контексте Ssonsvr.exe, когда сам Ssonsvr.exe не обращается к профилям в реестре? Работники службы поддержки Microsoft обратились за ответом к разработчикам Logical Prefetcher. Те прежде всего отметили, что реестр загружается в память Windows XP с использованием операций кешируемого ввода-вывода. Это означает, что поток упреждающего чтения заранее читает части кустов. Поскольку этот поток работает в процессе System, а Logical Prefetcher связывает активность процесса System с процессом, работающим в данный момент, определенная последовательность запуска и действия процессов во время загрузки и входа в систему могут привести к тому, что Logical Prefetcher припишет обращение к кусту реестра процессу Ssonsvr.exe. Если во время следующей загрузки и входа в систему порядок событий будет иным, Winlogon будет конфликтовать с Logical Prefetcher, что и видно в журнале загрузки.

Оказывается, вопреки его предназначению, операции Logical Prefetcher могут приводить к нарушениям совместного доступа, как в данном случае с Windows XP (в серверных ОС Logical Prefetcher проводит упреждающее чтение только для операций во время загрузки, и делает это синхронно, до начала загрузки). По этой причине в системах под управлением Windows Vista и Windows 7 Logical Prefetcher использует мини-драйвер фильтра файловой системы Fileinfo (%SystemRoot%\System32\Drivers\Fileinfo.sys). Этот драйвер отслеживает возможные нарушения совместного доступа и устраняет риск неудачи последующих операций с файлом, прочитанным Logical Prefetcher, пока последний не закроет этот файл.

Когда ситуация прояснилась, Microsoft и Citrix принялись усиленно искать временные решения для клиентов. Одним из вариантов стало отключение Prefetcher, а другим — написание сценария выхода, удаляющего файлы Prefetcher для Ssonsvr.exe. Компания Citrix опубликовала эти решения в статье Citrix Knowledge Base[3], а компания Microsoft — в статье № 969100 Microsoft Knowledge Base (*http://support.microsoft.com/kb/969100*).

---

[3] *http://support.citrix.com/artide/CTX118226*

Обновление клиента ICA, появившееся несколько дней спустя, откладывало загрузку DLL провайдера на 10 секунд после запуска Ssonsvr.exe, т.е. делало это до возврата управления Mpnotify.exe. Поскольку Winlogon ожидает завершения Mpnotify перед входом пользователя в систему, Logical Prefetcher не связывает обращение Winlogon к кусту реестра пользователя с загрузкой Ssonsvr.exe.

Как сказано выше, этот случай показался мне особенно интересным, поскольку он демонстрирует применение малоизвестной функции Procmon: журнал загрузки, а также возможности трассировки стеков для анализа первопричины проблемы — ключевые инструменты для поиска и устранения сбоев. Кроме того, из этого примера видно, как важны порой бывают временные решения, если средств решения проблемы нет или приходится ждать, пока производитель предоставит его. Вот так с помощью Procmon была решена очередная проблема!

Глава 17

Зависание и плохая производительность

В этой главе описаны случаи зависания приложений и низкой производительностью системы. Для решения проблем в этих случаях использовались, в основном, средства анализа стека вызовов, предоставляемые Procexp, Procmon и ProcDump.

- **Случай с IExplore, перегружавшем процессор**, демонстрирует использование стеков потоков в Procexp для выявления причины проблемы.
- **Случай со сбоем в ReadyBoost** описывает использование Procexp для выдвижения гипотезы о причинах проблемы и Procmon для ее подтверждения.
- **Случай с медленной демонстрацией** — еще одно подтверждение известного правила: если что-то может пойти не так, оно непременно пойдет не так, а вероятность неудачи при демонстрации пропорциональна численности аудитории. Трассировка Procmon позволила найти причину этой проблемы.
- **Случай медленного открытия файлов Project** демонстрирует использование окна Procmon **File Summary (Сводка по файлам)**, которое помогает быстро найти файлы с наибольшим числом обращений и файлы, обращение к которым занимает больше всего времени. Дальнейший анализ стеков вызовов позволяет выявить модуль, вызывающий проблемы с производительностью.
- **Сложный случай с зависанием Outlook** — пара связанных между собой проблем, описанных службой поддержки Microsoft и разрешенных с помощью утилиты ProcDump, которую я написал специально для этой службы.

Случай с IExplore, перегружавшим процессор

Однажды, установив Adobe Reader и закрыв Internet Explorer, я заметил, что значок Procexp в области уведомлений («трее») показывает аномально высокое использование ЦП. Я навел указатель мыши на этот значок и увидел во всплывающей подсказке, что процесс Iexplore.exe потребляет 50% ресурсов

ЦП (рис. 17-1). Поскольку у меня была двухпроцессорная система, я предположил, что один из потоков этого процесса вошел в бесконечный цикл.

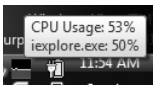


Рис. 17-1. Значок Procexp в области уведомлений и сообщение о высокой загруженности ЦП процессом Iexplore.exe

Я открыл Procexp, нашел процесс Iexplore.exe, открыл окно его свойств и перешел на вкладку **Threads (Потоки)**. Как я и ожидал, один поток был привязан к процессору (рис. 17-2). Это демонстрирует одно из преимуществ многопроцессорных систем: максимум ресурсов, потребляемых вышедшим из-под контроля потоком, не может превышать времени одного процессора (т.е. 50% общих ресурсов двухпроцессорной системы), что оставляет достаточно ресурсов ЦП для других задач, включая решение возникшей проблемы. На однопроцессорной системе вышедший из-под контроля поток блокирует всю систему.

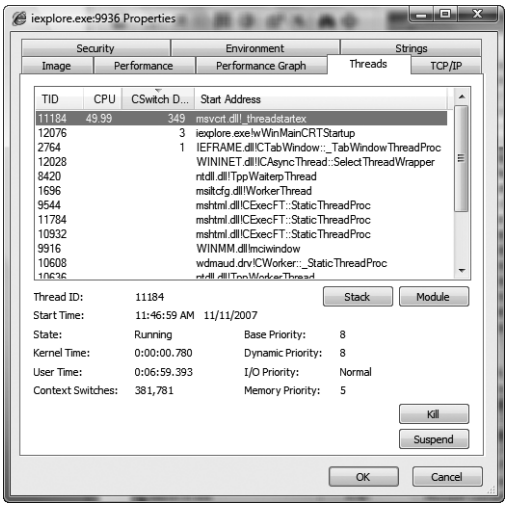


Рис. 17-2. Вышедший из-под контроля поток перегрузил один из процессоров в двуядерной системе

Стартовый адрес потока, вышедшего из-под контроля, не дал никаких намеков, это был просто стандартная точка входа потока в DLL исполняющей среды Windows C. Чтобы понять, какой код он запускал, я выбрал его в списке потоков и щелкнул на кнопке **Stack (Стек)**. В стеке вызовов был показан код, полученный из gp.ocx (строки 21-25 на рис. 17-3).

Я никогда не слышал о gp.ocx, поэтому я открыл представление DLL и начал искать его в процессе Iexplore.exe. Этот файл представился как «getPlus(R) ActiveX Control» компании NOS Microsystems Ltd (рис. 17-4).

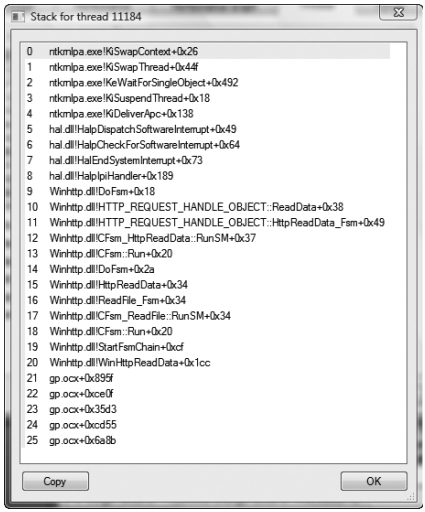


Рис. 17-3. Код вышедшего из-под контроля потока получен из gp.ocx

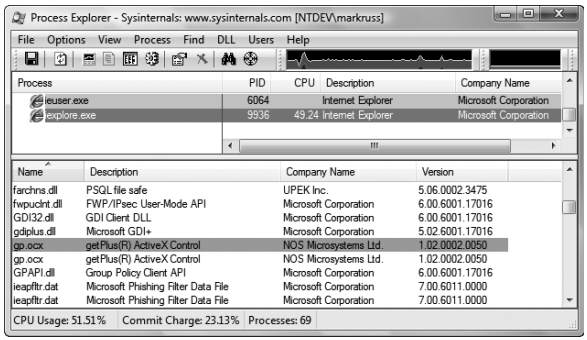


Рис. 17-4. Поиск gp.ocx в представлении DLL

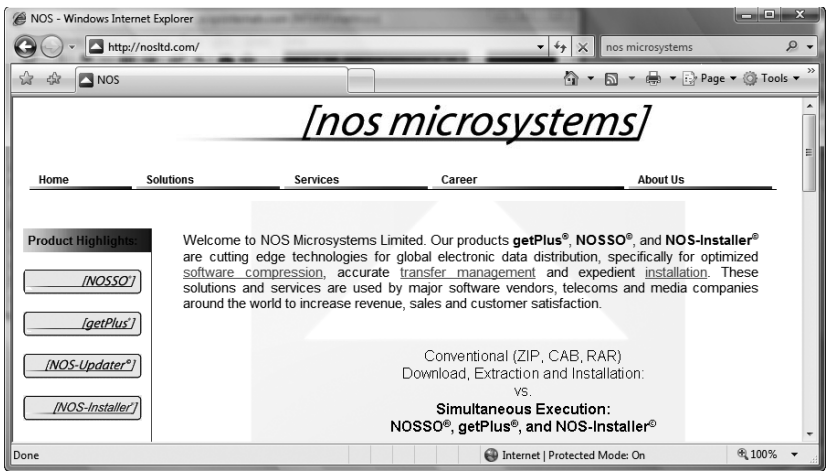


Рис. 17-5. Веб-страница NOS Microsystems

Я выполнил поиск «NOS Microsystems» в поисковой системе Bing и нашел веб-страницу этой компании (рис. 17-5). Она выглядела обычной фирмой, и я смутно помнил, что видел название «getPlus» в программе загрузки Adobe Reader. Я запустил Autoruns и убедился, что gp.ocx не запускается автоматически, а загружается только в том случае, если веб-страница явно вызывает этот файл, что я счел маловероятным, после чего завершил Iexplore.exe в Procexp и перезапустил Internet Explorer. Убедившись, что gp.ocx не загружен, я закрыл это дело.

Дело о проблеме с ReadyBoost

Пользователь работал на лэптопе с Windows 7 больше года, не имея проблем, часто не включая его неделями. Однако потом появились проблемы с выходом из спящего режима. Производительность была низкой, а индикатор работы жесткого диска горел без перерыва минимум пять минут.

Он запустил Procexp, чтобы посмотреть, какой процесс или процессы потребляют время ЦП, и обнаружил, что процесс System занимает около 35%, что много для двухпроцессорной системы. Он дважды щелкнул имя этого процесса, чтобы открыть диалоговое окно свойств, и перешел на вкладку **Threads (Потоки)**, где увидел, что источник проблемы имеет стартовый адрес в Rdyboost.sys, драйвере ReadyBoost (рис. 17-6).

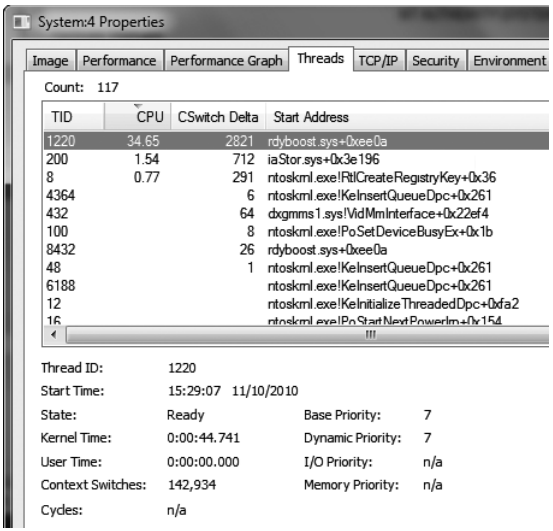


Рис. 17-6. Поток System, начинающийся в Rdyboost.sys, потребляет 35% времени ЦП

ReadyBoost — функция Windows Vista и Windows 7, повышающая производительность, используя твердотельные накопители, такие как карты SD и USB Thumb Drive, в качестве кеша памяти.

Чтобы убедиться, что проблема именно в ReadyBoost, он запустил трассировку Procmon. Сначала он не увидел ничего интересного, но затем вспом-

нил, что надо удалить стандартный фильтр, скрывающий активность процесса System (рис. 17-7).

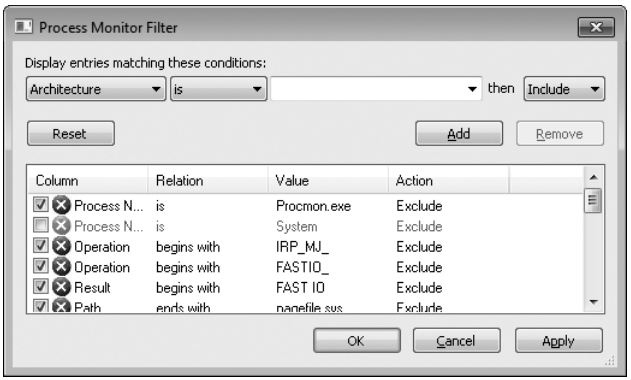


Рис. 17-7. Удаление фильтра Exclude для отображения активности процесса System

Трассировка показала длительные серии операций чтения с диска H — флеш-карты емкостью 8 Гб, настроенной для использования с ReadyBoost (рис. 17-8).

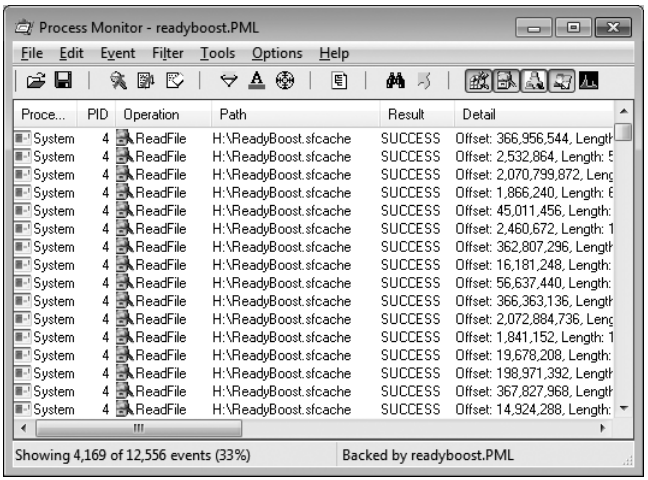


Рис. 17-8. Длительные серии операций чтений из кеша ReadyBoost на диске H

Наконец, он просмотрел сводку по файлам (**File Summary**) и обнаружил, что большая часть процессорного времени тратится на чтение с диска ReadyBoost (рис. 17-9). Поняв, в чем причина проблемы, он извлек флеш-карту, и компьютер сразу стал работать нормально. Подобные проблемы с ReadyBoost возникают редко. Вероятно, это было вызвано каким-то параметром конфигурации или плохой флеш-картой.

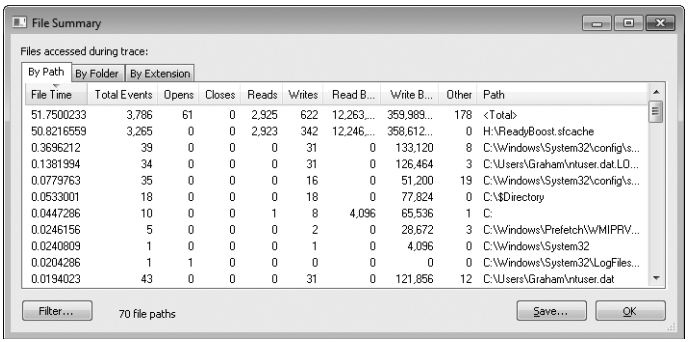


Рис. 17-9. Окно Procmon File Summary показывает, что много времени тратится на чтение из кеша ReadyBoost

Случай с медленной демонстрацией

В 2009 г. я выступал на конференции Microsoft TechEd US с аудиторией более 5000 человек[1]. Билл Вехт, главный вице-президент отдела маркетинга Windows, рассказывал о функциях Windows 7, ориентированных на пользователя, Лэйн Макдональд, главный менеджер Windows Server, представлял новинки Hyper-V и Windows Server 2008 R2, а я демонстрировал улучшения в Windows 7 и Microsoft Desktop Optimization Pack (MDOP), ориентированные на ИТ-специалистов.

Я показывал такие функции, как параметры групповой политики BitLocker-To-Go, возможности удаленной работы с PowerShell 2, возможности PowerShell по созданию сценариев с объектами групповой политики, Microsoft Enterprise Desktop Virtualization (MED-V), а также рассказывал, как объединение App-V, перемещаемых профилей пользователей и перенаправления папок позволяет заменять системы с минимальным временем простоя. Я подчеркивал тот факт, что мы приложили все усилия, чтобы исправления, обеспечивающие совместимость приложений (так называемые прокладки, shims), разработанные для Windows Vista, работали и в Windows 7. Также я демонстрировал новую функцию Windows 7 AppLocker, позволяющую ограничивать программы, разрешенные для запуска на корпоративных компьютерах, с гибкими правилами идентификации ПО.

За несколько недель до конференции я работал с Джейсоном Лезнеком, ответственным за секцию для ИТ-специалистов, над функциями, которые я буду освещать, и над проектом демонстрационных материалов. Мы прогоняли сценарий, подправляли демонстрационные материалы и работали над переходами, урезая содержимое, чтобы уложиться в отведенное мне время и сжать мой доклад, сфокусировав его на преимуществах новых технологий. Для демонстрации совместимости приложений мы решили использо-

[1] Наше выступлении доступно для просмотра онлайн: *http:// http://www.msteched.com/2009/NorthAmerica/KEY01*. Моя часть начинается примерно с 42 мин.

вать образец программы StockViewer, созданной Крисом Джексоном (специалистом по совместимости приложений) для демонстрации распространенных ошибок, вызывающих проблемы совместимости в Windows Vista и Windows 7. (Сейчас StockViewer — демонстрационное приложение, входящее в состав Microsoft Application Compatibility Toolkit.) В моей демонстрации я запускал StockViewer в Windows 7 и показывал, как его функция Trends завершается с неясным сообщением об ошибке, вызванной проблемами с совместимостью (рис. 17-10). Затем я собирался показать, как можно установить «прокладку», обеспечивающую совместимость и позволяющую приложению нормально работать в Windows Vista, после чего успешно запускал приложение.

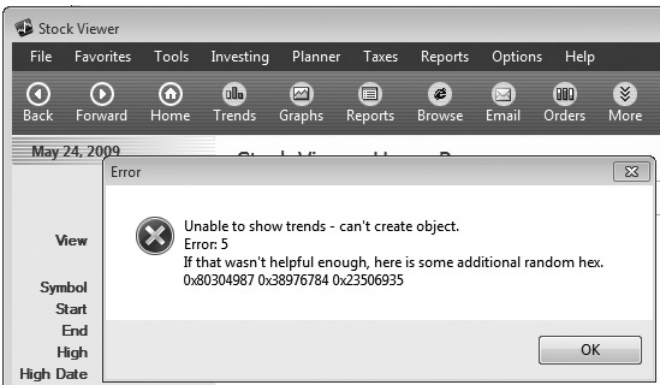


Рис. 17-10. Ошибка StockViewer, вызванная проблемами с совместимостью

Также мы хотели показать, как с помощью мастера создания правил AppLocker можно быстро разрешить запуск приложений, основываясь на издателе или версии, если эти приложения имеют цифровые подписи. Изначально мы планировали показывать AppLocker после демонстрации совместимости приложений, используя распространенное приложение Adobe Acrobat Reader. Мы прорепетировали этот сценарий несколько раз, но сочли переходы не очень удачными, и я предложил подписать исполняемый файл StockViewer и сначала продемонстрировать AppLocker, а потом «прокладку». Тогда я бы смог разрешить запуск StockViewer при помощи правила AppLocker, а затем показать, как «прокладка» обеспечивает его корректную работу.

Я вернулся в офис, подписал StockViewer сертификатом Sysinternals и отправил его Джейсону. Через несколько часов он прислал мне письмо, в котором говорил, что с демонстрацией что-то не так: запуск StockViewer, ранее выполнявшийся мгновенно, теперь требует больше минуты ожидания. Сроки поджимали, и Джейсон запаниковал, потому что нам надо было закончить подготовку демонстраций. Я когда-то слышал, что .NET выполняет проверку подписей Authenticode, когда загружает подписанные сборки, поэтому первое подозрение пало именно на это. Я попросил Джейсона сделать трассировку в Process Monitor, и он прислал ее через несколько минут.

Открыв журнал, первым делом я отфильтровал события, относящиеся к StockViewer.exe, найдя его первую операцию и щелкнув ее правой кнопкой, чтобы быстро установить фильтр (рис. 17-11).

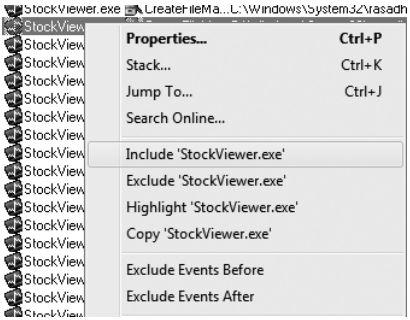


Рис. 17-11. Установка фильтра для StockView.exe

Затем я просмотрел временные отметки первого (2:27:20) и последнего (2:28:32) элементов, что соответствовало минутной задержке, о которой писал Джейсон. В результатах трассировки я увидел много ссылок на криптографические разделы реестра и папки файловой системы, а также ссылки на параметры TCP/IP, но я знал, что должен быть минимум один промежуток, соответствующий длительной задержке. Я просмотрел журнал с начала и обнаружил промежуток примерно в 10 секунд (рис. 17-12).

Time of Day	Process Name	Operation	Path	Result
2:27:21.1500415 PM	StockViewer.exe	CloseFile	C:\Windows\System32\rasadhlp.dll	SUCCESS
2:27:21.1501741 PM	StockViewer.exe	CreateFile	C:\Windows\System32\rasadhlp.dll	SUCCESS
2:27:21.1503016 PM	StockViewer.exe	CreateFileMa...	C:\Windows\System32\rasadhlp.dll	FILE LOCKED WI...
2:27:21.1503485 PM	StockViewer.exe	CreateFileMa...	C:\Windows\System32\rasadhlp.dll	SUCCESS
2:27:21.1505176 PM	StockViewer.exe	Load Image	C:\Windows\System32\rasadhlp.dll	SUCCESS
2:27:21.1505375 PM	StockViewer.exe	CloseFile	C:\Windows\System32\rasadhlp.dll	SUCCESS
2:27:22.0634600 PM	StockViewer.exe	Thread Create		SUCCESS
2:27:32.3219796 PM	StockViewer.exe	CreateFile	C:\Windows\System32\en-US\winhttp.dll.mui	SUCCESS
2:27:32.3220801 PM	StockViewer.exe	CreateFileMa...	C:\Windows\System32\en-US\winhttp.dll.mui	FILE LOCKED WI...
2:27:32.3220976 PM	StockViewer.exe	QueryStanda...	C:\Windows\System32\en-US\winhttp.dll.mui	SUCCESS
2:27:32.3221374 PM	StockViewer.exe	CreateFileMa...	C:\Windows\System32\en-US\winhttp.dll.mui	SUCCESS
2:27:32.3224601 PM	StockViewer.exe	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Services\crypt32	REPARSE
2:27:32.3225041 PM	StockViewer.exe	RegOpenKey	HKLM\System\CurrentControlSet\Services\crypt32	SUCCESS
2:27:32.3225577 PM	StockViewer.exe	RegQueryVa...	HKLM\System\CurrentControlSet\Services\crypt32\Debu...	NAME NOT FOUND

Рис. 17-12. 10-секундный промежуток между событиями StockViewer

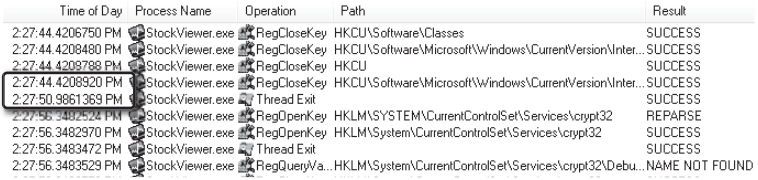
Time of Day	Process Name	Operation	Path	Result
2:27:32.4735745 PM	StockViewer.exe	RegCloseKey	HKCU	SUCCESS
2:27:32.4735906 PM	StockViewer.exe	RegCloseKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Inter...	SUCCESS
2:27:32.4741019 PM	StockViewer.exe	RegQueryVa...	HKLM\System\CurrentControlSet\Services\WinSock2\P...	SUCCESS
2:27:32.4741331 PM	StockViewer.exe	RegOpenKey	HKLM\System\CurrentControlSet\Services\WinSock2\P...	NAME NOT FOUND
2:27:44.3248702 PM	StockViewer.exe	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Services\crypt32	REPARSE
2:27:44.3249124 PM	StockViewer.exe	RegOpenKey	HKLM\System\CurrentControlSet\Services\crypt32	SUCCESS
2:27:44.3249669 PM	StockViewer.exe	RegQueryVa...	HKLM\System\CurrentControlSet\Services\crypt32\Debu...	NAME NOT FOUND
2:27:44.3249930 PM	StockViewer.exe	Thread Exit		SUCCESS
2:27:44.3249967 PM	StockViewer.exe	RegCloseKey	HKLM\System\CurrentControlSet\Services\crypt32	SUCCESS
2:27:44.3252962 PM	StockViewer.exe	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Services\crypt32	REPARSE
2:27:44.3253251 PM	StockViewer.exe	RegOpenKey	HKLM\System\CurrentControlSet\Services\crypt32	SUCCESS

Рис. 17-13. 12-секундный промежуток между событиями StockViewer

Непосредственно перед этим были ссылки на библиотеку Rasadhlp.dll, связанную с работой в сети, немного раньше — много обращений к разделам реестра Winsock, а сразу после задержки — обращения к криптографическим разделам реестра. Похоже, что компьютер не была подключен к Интернету,

и приложение ждало истечения таймаута какого-то сетевого компонента около 10 секунд. Я продолжил просмотр и нашел 12-секундный интервал (рис. 17-13).

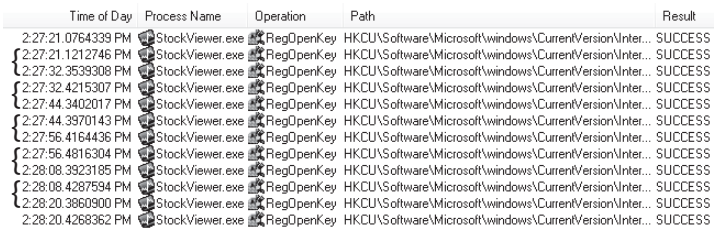
Перед этим промежутком снова была сетевая активность и «криптографическая» после. Следующий промежуток тоже длился 12 секунд (рис. 17-14).



Time of Day	Process Name	Operation	Path	Result
2:27:44.4206750 PM	StockViewer.exe	RegCloseKey	HKCU\Software\Classes	SUCCESS
2:27:44.4208480 PM	StockViewer.exe	RegCloseKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Inter...	SUCCESS
2:27:44.4209788 PM	StockViewer.exe	RegCloseKey	HKCU	SUCCESS
2:27:44.4208920 PM	StockViewer.exe	RegCloseKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Inter...	SUCCESS
2:27:50.9861369 PM	StockViewer.exe	Thread Exit		SUCCESS
2:27:56.3482524 PM	StockViewer.exe	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Services\crypt32	REPARSE
2:27:56.3482970 PM	StockViewer.exe	RegOpenKey	HKLM\System\CurrentControlSet\Services\crypt32	SUCCESS
2:27:56.3483472 PM	StockViewer.exe	Thread Exit		SUCCESS
2:27:56.3483529 PM	StockViewer.exe	RegQueryVa...	HKLM\System\CurrentControlSet\Services\crypt32\Debu...	NAME NOT FOUND

Рис. 17-14. Еще один 12-секундный промежуток между событиями

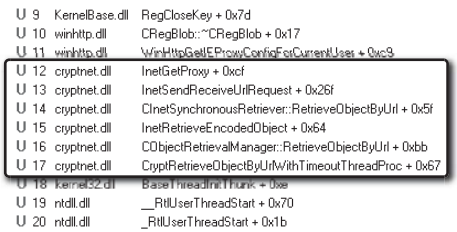
Следующие несколько промежутков были практически идентичными. В каждом случае непосредственно перед паузой было обращение к HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections, поэтому я установил фильтр для этого пути и для операции *RegOpenKey* и, конечно же, нашел пять промежутков по 12 секунд каждый (рис. 17-15).



Time of Day	Process Name	Operation	Path	Result
2:27:21.0764339 PM	StockViewer.exe	RegOpenKey	HKCU\Software\Microsoft\windows\CurrentVersion\Inter...	SUCCESS
2:27:21.1212746 PM	StockViewer.exe	RegOpenKey	HKCU\Software\Microsoft\windows\CurrentVersion\Inter...	SUCCESS
2:27:32.3539308 PM	StockViewer.exe	RegOpenKey	HKCU\Software\Microsoft\windows\CurrentVersion\Inter...	SUCCESS
2:27:32.4215307 PM	StockViewer.exe	RegOpenKey	HKCU\Software\Microsoft\windows\CurrentVersion\Inter...	SUCCESS
2:27:44.3402017 PM	StockViewer.exe	RegOpenKey	HKCU\Software\Microsoft\windows\CurrentVersion\Inter...	SUCCESS
2:27:44.3970143 PM	StockViewer.exe	RegOpenKey	HKCU\Software\Microsoft\windows\CurrentVersion\Inter...	SUCCESS
2:27:56.4164436 PM	StockViewer.exe	RegOpenKey	HKCU\Software\Microsoft\windows\CurrentVersion\Inter...	SUCCESS
2:27:56.4816304 PM	StockViewer.exe	RegOpenKey	HKCU\Software\Microsoft\windows\CurrentVersion\Inter...	SUCCESS
2:28:08.3923185 PM	StockViewer.exe	RegOpenKey	HKCU\Software\Microsoft\windows\CurrentVersion\Inter...	SUCCESS
2:28:08.4287594 PM	StockViewer.exe	RegOpenKey	HKCU\Software\Microsoft\windows\CurrentVersion\Inter...	SUCCESS
2:28:20.3860900 PM	StockViewer.exe	RegOpenKey	HKCU\Software\Microsoft\windows\CurrentVersion\Inter...	SUCCESS
2:28:20.4268362 PM	StockViewer.exe	RegOpenKey	HKCU\Software\Microsoft\windows\CurrentVersion\Inter...	SUCCESS

Рис. 17-15. Пять 12-секундных промежутков

В сумме эти промежутки (5 по 12) давали задержку, возникающую у Джейсона. Теперь я хотел убедиться, что повторные попытки доступа к сети были вызваны проверкой подписи, так что я стал просматривать стеки вызовов различных событий, выбирая их и нажимая Ctrl+K для вызова окна свойств стека. Стек событий, относящихся к параметрам подключения к Интернету, показал, что причиной были криптографические функции (рис. 17-16).



U	9	KernelBase.dll	RegCloseKey + 0x7d
U	10	winhttp.dll	CRegBlob::~CRegBlob + 0x17
U	11	winhttp.dll	WinHttpGetIEProxyConfigForCurrentUser + 0xc9
U	12	cryptnet.dll	InetGetProxy + 0xcf
U	13	cryptnet.dll	InetSendReceiveUrlRequest + 0x26f
U	14	cryptnet.dll	CInetSynchronousRetriever::RetrieveObjectByUrl + 0x5f
U	15	cryptnet.dll	InetRetrieveEncodedObject + 0x64
U	16	cryptnet.dll	CObjectRetrievalManager::RetrieveObjectByUrl + 0xbb
U	17	cryptnet.dll	CryptRetrieveObjectByUrlWithTimeoutThreadProc + 0x67
U	18	kernel32.dll	BaseThreadInitThunk + 0xe
U	19	ntdll.dll	__RtlUserThreadStart + 0x70
U	20	ntdll.dll	_RtlUserThreadStart + 0x1b

Рис. 17-16. Стек вызовов показал причастность криптографических операций

Наконец, я хотел убедиться, что эти проверки были вызваны средой .NET. Я еще раз просмотрел журнал и увидел события, подтверждающие, что StockViewer — приложение .NET (рис. 17-17).

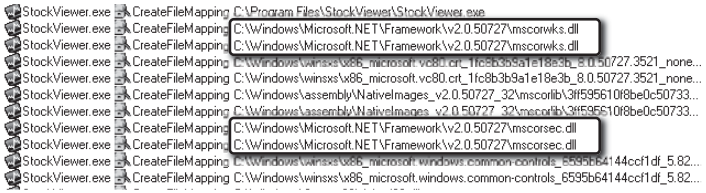


Рис. 17-17. Доказательство причастности .NET

Также я просмотрел стеки некоторых предшествующих событий, ссылающихся на «криптографические» разделы реестра, и увидел, что исполняющая среда .NET вызывала *WinVerifyTrust* — функцию Windows для проверки цифровой подписи файла, запускающую серию попыток выхода в Интернет (рис. 17-18).

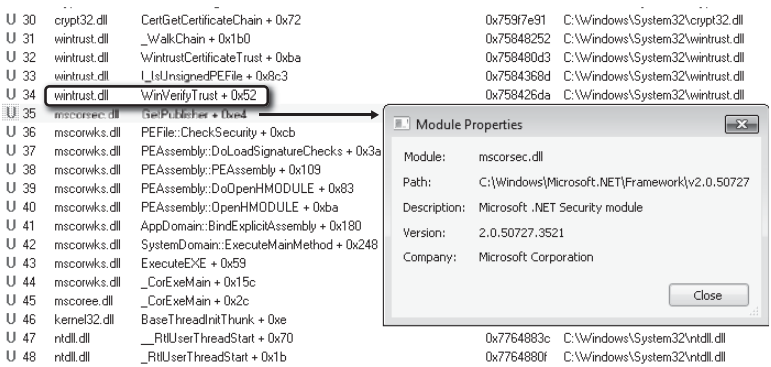


Рис. 17-18. Вызов ***WinVerifyTrust*** из .NET Framework

Теперь я точно знал причину задержки: среда .NET обнаруживала, что файл Stockviewer.exe имеет цифровую подпись, и пыталась проверить, не отозван ли сертификат. Я стал искать в Интернете способ пропустить эту проверку .NET, потому что знал, что компьютеры на конференции, скорее всего, не будут подключены к Интернету во время нашего выступления. Через несколько минут поиска я нашел статью Knowledge Base 936707 «FIX: A .NET Framework 2.0 managed application that has an Authenticode signature takes longer than usual to start» («Исправление: управляемое приложение .NET Framework 2.0, имеющее подпись Authenticode, запускается дольше обычного») (*http://support.microsoft.com/kb/936707*). В этой статье описывались точно те же симптомы, что и у нас, и говорилось, что .NET 2.0 (а, судя по путям к библиотекам DLL, указанным в данных трассировки, в StockViewer использовалась именно эта версия) поддерживает способ отключения обязательной проверки цифровых подписей сборок. Для этого нужно создать в папке исполняемого файла файл конфигурации с тем же именем, что и у

исполняемого файла, но с расширением .config (например, StockViewer.exe.config), содержащий следующий код XML:

```
<?xml version=»1.0» encoding=»utf-8»?>
<configuration>
      <runtime>
               <generatePublisherEvidence enabled=»false»/>
      </runtime>
</configuration>
```

Примерно через 15 минут после получения письма от Джейсона я отправил ему ответ с объяснением и файлом конфигурации. Вскоре после этого он написал, что задержки исчезли, и выразил удивление тем, как быстро я решил проблему. Для него это казалось чем-то вроде магии, но я просто использовал базовые приемы решения проблем с использованием Procmon и Интернета. И, конечно, демонстрация прошла на ура.

Случай с медленным открытием файлов Project

Все началось, когда клиент, сетевой администратор, обратился в службу поддержки Microsoft с жалобой пользователя на то, что файлы Microsoft Project, расположенные в общей папке в сети, открываются около минуты, и часто появляется ошибка, показанная на рис. 17-19.

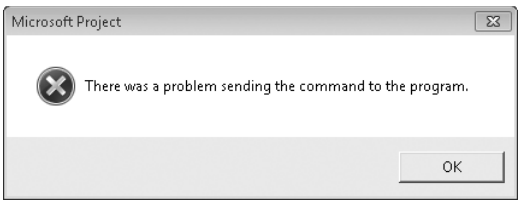


Рис. 17-19. Ошибка, возникающая при открытии файлов Project

Администратор убедился в наличии такой проблемы, проверил параметры сети и задержки на файловом сервере, но не смог обнаружить ничего, что объясняло бы происходящее. Инженер службы поддержки попросил сделать трассировку Procmon и Network Monitor при открытии проблемного файла. После получения данных он открыл журнал Procmon и установил фильтр, включающий только операции процесса Project, и еще один фильтр, включающий пути, ссылающиеся на целевую общую папку. В столбце **File Time (Время файла)** диалогового окна **File Summary (Сводка по файлам)**, открытого из меню **Tools (Сервис)**, было показано, что на обращение к файлам из общей папки тратится много времени (рис. 17-20).

Пути из данных трассировки показывали, что профили пользователей хранились на файловом сервере, и запуск Project приводил к активному обращению к каталогу AppData профиля. Если многие пользователи хранили свои профили на одном сервере, используя перенаправление папок, и запускали одинаковые приложения, хранящие данные в папке AppData, это,

определенно, приводило к некоторой задержке. Давно известно, что перенаправление каталога AppData может приводить к ухудшению производительности, поэтому первая инструкция инженера службы поддержки была следующей: настроить перемещаемые профили пользователей так, чтобы папка AppData не перенаправлялась и синхронизировалась только при входе и выходе из системы, в соответствии с рекомендациями из блога Microsoft[2]:

Особые соображения насчет папки AppData\Roaming:

Если папка AppData\Roaming перенаправляется, производительность некоторых приложений может ухудшаться, так как они будут обращаться к этой папке по сети. В таком случае рекомендуется настроить следующий параметр групповой политики для синхронизации папки AppData\Roaming только при входе и выходе из системы и использования локального кеша во время сеанса работы пользователя. Это может замедлить вход и выход из системы, но приложение будет работать быстрее, поскольку не будет ожидать ответа из сети:

User configuration > Administrative Templates > System > User Profiles > сетевые_папки > настройка для синхронизации только при входе и выходе.

Если проблемы с приложениями остаются, следует отменить перенаправление AppData. Недостатком этого метода может быть замедление входа и выхода из системы.

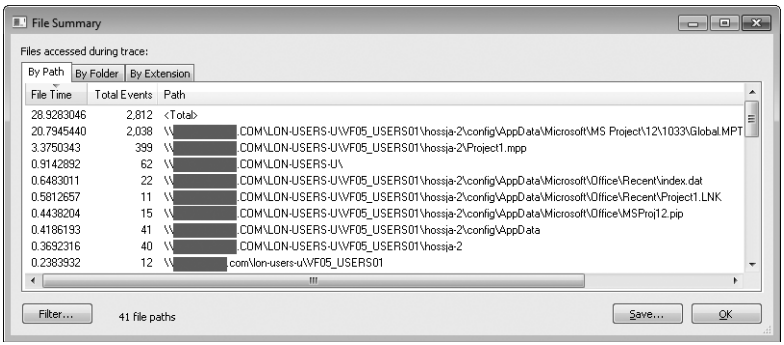


Рис. 17-20. Окно File Summary показывает, что на операции с файлами тратится много времени (имя домена скрыто)

Затем инженер проверил, генерирует ли Project весь трафик к таким файлам, как Global.MPT, или это связано с дополнениями. Для этого понадобились данные трассировки стеков. Установив фильтр, отображающий только обращения к Global.MPT (файлу с наибольшим временем ввода-вывода, как видно из окна сводки), он заметил, что тот открывался и читался несколько раз. Сначала он увидел пять или шесть длительных серий из коротких случайных операций чтения (рис. 17-21).

---

[2] «Профили пользователей служб удаленного рабочего стола Windows Server 2008 R2», *http://blogs.msdn.com/b/rds/archive/2009/06/02/user-profiles-on-windows-server-2008-r2-remote-desktop-services.aspx*

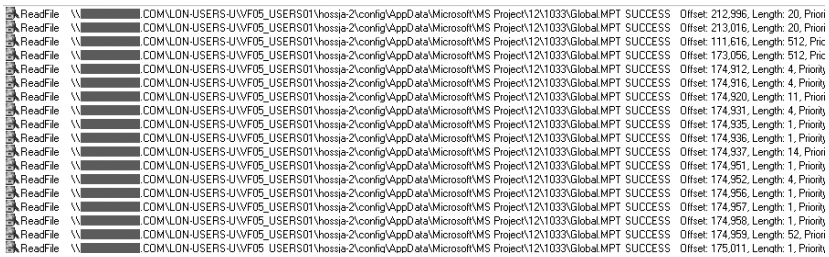


Рис. 17-21. Длительные серии коротких произвольных операций чтения по сети

Однако стеки для этих операций показывали, что дело в самом Project. На рис. 17-22 во фрейме 25 показан модуль WINPROJ.EXE, вызывающий код из Ole32.dll, который вызывает Kernel32.dll (фрейм 15), а тот вызывает API-функцию *ReadFile* из Kernelbase.dll (все это DLL Windows).

Frame	Module	Location
U 6	wow64cpu.dll	CpupSyscallStub + 0x9
U 7	wow64cpu.dll	ReadWriteFileFault + 0x31
U 8	wow64.dll	RunCpuSimulation + 0xa
U 9	wow64.dll	Wow64LdrpInitialize + 0x429
U 10	ntdll.dll	LdrpInitializeProcess + 0x17e2, d:\
U 11	ntdll.dll	_LdrpInitialize + 0x14533
U 12	ntdll.dll	LdrInitializeThunk + 0xe, d:\w7rtm
U 13	ntdll.dll	ZwReadFile + 0x15, o:\w7rtm.obj
U 14	KERNELBASE.dll	ReadFile + 0x118
U 15	kernel32.dll	ReadFileImplementation + 0xf0
U 16	ole32.dll	CFileStream::ReadAt_FromFile + 0
U 17	ole32.dll	CFileStream::ReadAt + 0xb1, d:\w
U 18	ole32.dll	CDirectStream::ReadAt + 0x222, c
U 19	ole32.dll	CDirectStream::ReadAt + 0x1e7, c
U 20	ole32.dll	PSStream::ReadAt + 0x46, d:\w7r
U 21	ole32.dll	CTransactedStream::ReadAt + 0x
U 22	ole32.dll	PSStream::ReadAt + 0x3f, d:\w7rt
U 23	ole32.dll	CPubStream::ReadAt + 0x56, d:\w
U 24	ole32.dll	CExposedStream::Read + 0x7d, d
U 25	WINPROJ.EXE	WINPROJ.EXE + 0x123663
U 26	WINPROJ.EXE	WINPROJ.EXE + 0x126528

Рис. 17-22. Winproj.exe вызывает код Windows для чтения файла

Также инженер увидел серии некешируемых операций чтения больших объемов данных (рис. 17-23). Небольшие операции, которые он увидел сначала, кешировались, так что после первого обращения доступ по сети не повторялся. Но некешированные данные каждый раз читались с сервера, что делало их более вероятной причиной снижения производительности.

Рис. 17-23. Серии операций чтения больших объемов некешированных данных по сети

Ухудшало ситуацию то, что один и тот же файл считывался по сети несколько раз. На рис. 17-24 показаны отфильтрованные данные о первоначальных операциях чтения файлов, в которых смещение в столбце **Detail (Подробно)** равно 0.

Operation	Path	Result	Detail
ReadFile	\\\\[redacted].COM\LON-USE...	SUCCESS	Offset: 0, Length: 4,096, I/O Flags: Non-cached, Paging I/O, Priority: Normal
ReadFile	\\\\[redacted].COM\LON-USE...	SUCCESS	Offset: 0, Length: 4,096, I/O Flags: Non-cached, Paging I/O, Priority: Normal
ReadFile	\\\\[redacted].COM\LON-USE...	SUCCESS	Offset: 0, Length: 4,096, I/O Flags: Non-cached, Paging I/O, Priority: Normal
ReadFile	\\\\[redacted].COM\LON-USE...	SUCCESS	Offset: 0, Length: 4,096, I/O Flags: Non-cached, Paging I/O, Priority: Normal
ReadFile	\\\\[redacted].COM\LON-USE...	SUCCESS	Offset: 0, Length: 4,096, I/O Flags: Non-cached, Paging I/O, Priority: Normal
ReadFile	\\\\[redacted].COM\LON-USE...	SUCCESS	Offset: 0, Length: 4,096, I/O Flags: Non-cached, Paging I/O, Priority: Normal

Рис. 17-24. Файлы повторно считываются по сети; смещение 0 указывает на то, что чтение производится с начала файла

Стеки для этих операций чтения показали, что они выполняются драйвером стороннего производителя STRSP64.SYS. Первое указание на то, что это сторонний драйвер, можно увидеть во фреймах 18-21 окна с данными трассировки стека (рис. 17-25). Procmon настроен для получения символов с серверов Microsoft, а драйвер SRTSP64.SYS не имеет информации о символах и вызывает *FltReadFile* (фрейм 17).

Frame	Module	Location	Path
K 0	fltmgr.sys	FltpPerformPreCallbacks + ...	C:\Windows\system32\drivers\fltmgr.sys
K 1	fltmgr.sys	FltpPassThrough + 0x2d9	C:\Windows\system32\drivers\fltmgr.sys
K 2	fltmgr.sys	FltpDispatch + 0xb7	C:\Windows\system32\drivers\fltmgr.sys
K 3	ntoskrnl.exe	IoPageRead + 0x252	C:\Windows\system32\ntoskrnl.exe
K 4	ntoskrnl.exe	MiPfExecuteReadList + 0xff	C:\Windows\system32\ntoskrnl.exe
K 5	ntoskrnl.exe	MmPrefetchForCacheMan...	C:\Windows\system32\ntoskrnl.exe
K 6	ntoskrnl.exe	CcFetchDataForRead + 0x...	C:\Windows\system32\ntoskrnl.exe
K 7	ntoskrnl.exe	CcCopyRead + 0x16b	C:\Windows\system32\ntoskrnl.exe
K 8	rdbss.sys	RxCommonRead + 0xdb1, ...	C:\Windows\system32\DRIVERS\rdbss.sys
K 9	rdbss.sys	RxFsdCommonDispatch + ...	C:\Windows\system32\DRIVERS\rdbss.sys
K 10	rdbss.sys	RxFsdDispatch + 0x224, d...	C:\Windows\system32\DRIVERS\rdbss.sys
K 11	mrxsmb.sys	MRxSmbFsdDispatch + 0x...	C:\Windows\system32\DRIVERS\mrxsmb.sys
K 12	mup.sys	MupiCallUncProvider + 0x...	C:\Windows\System32\Drivers\mup.sys
K 13	mup.sys	MupStateMachine + 0x12...	C:\Windows\System32\Drivers\mup.sys
K 14	mup.sys	MupFsdIrpPassThrough + ...	C:\Windows\System32\Drivers\mup.sys
K 15	fltmgr.sys	FltpLegacyProcessingAfter...	C:\Windows\system32\drivers\fltmgr.sys
K 16	fltmgr.sys	FltPerformSynchronousIo +...	C:\Windows\system32\drivers\fltmgr.sys
K 17	fltmgr.sys	FltReadFile + 0x334	C:\Windows\system32\drivers\fltmgr.sys
K 18	SRTSP64.SYS	SRTSP64.SYS + 0x2b11b	C:\Windows\System32\Drivers\SRTSP64.SYS
K 19	SRTSP64.SYS	SRTSP64.SYS + 0x3c49f	C:\Windows\System32\Drivers\SRTSP64.SYS
K 20	SRTSP64.SYS	SRTSP64.SYS + 0x691b6	C:\Windows\System32\Drivers\SRTSP64.SYS
K 21	SRTSP64.SYS	SRTSP64.SYS + 0x69241	C:\Windows\System32\Drivers\SRTSP64.SYS

Рис. 17-25. Srtsp64.sys в стеке вызовов первоначальных операций чтения файлов

Далее, фреймы, стоящие выше в том же стеке (рис. 17-26), показали, что серии операций чтения SRTSP64.SYS шли в контексте обратных вызовов диспетчера фильтров (фрейм 31), выполнявшихся, когда файл открывался в Project вызовом *CreateFileW* (фрейм 50). Такое поведение обычно для антивирусов, выполняющих проверку при обращении к файлу.

Двойной щелчок одной из строк с SRTSP64.SYS в стеке открыл окно свойств модуля (рис. 17-27), которое подтвердило, что это файл Symantec AutoProtect, выполняющий проверку на вирусы каждый раз, когда файл Project открывается с определенными параметрами.

Frame	Module	Location	Path
K 30	SRTSP64.SYS	SRTSP64.SYS + 0x2052e	C:\Windows\System32\Drivers\SRTSP64.SYS
K 31	fltmgr.sys	FltpPerformPostCallbacks ...	C:\Windows\system32\drivers\fltmgr.sys
K 32	fltmgr.sys	FltpLegacyProcessingAfter...	C:\Windows\system32\drivers\fltmgr.sys
K 33	fltmgr.sys	FltpCreate + 0x2a9	C:\Windows\system32\drivers\fltmgr.sys
K 34	ntoskrnl.exe	IopParseDevice + 0x5a7	C:\Windows\system32\ntoskrnl.exe
K 35	ntoskrnl.exe	ObpLookupObjectName + ...	C:\Windows\system32\ntoskrnl.exe
K 36	ntoskrnl.exe	ObOpenObjectByName + ...	C:\Windows\system32\ntoskrnl.exe
K 37	ntoskrnl.exe	IopCreateFile + 0x2b7	C:\Windows\system32\ntoskrnl.exe
K 38	ntoskrnl.exe	NtCreateFile + 0x78	C:\Windows\system32\ntoskrnl.exe
K 39	ntoskrnl.exe	KiSystemServiceCopyEnd ...	C:\Windows\system32\ntoskrnl.exe
U 40	ntdll.dll	ZwCreateFile + 0xa, o:\w7...	C:\Windows\System32\ntdll.dll
U 41	wow64.dll	whNtCreateFile + 0x10f	C:\Windows\System32\wow64.dll
U 42	wow64.dll	Wow64SystemServiceEx ...	C:\Windows\System32\wow64.dll
U 43	wow64cpu.dll	TurboDispatchJumpAddre...	C:\Windows\System32\wow64cpu.dll
U 44	wow64.dll	RunCpuSimulation + 0xa	C:\Windows\System32\wow64.dll
U 45	wow64.dll	Wow64LdrpInitialize + 0x429	C:\Windows\System32\wow64.dll
U 46	ntdll.dll	LdrpInitializeProcess + 0x1...	C:\Windows\System32\ntdll.dll
U 47	ntdll.dll	_LdrpInitialize + 0x14533	C:\Windows\System32\ntdll.dll
U 48	ntdll.dll	LdrInitializeThunk + 0xe, d:...	C:\Windows\System32\ntdll.dll
U 49	ntdll.dll	NtCreateFile + 0x12, o:\w7...	C:\Windows\SysWOW64\ntdll.dll
U 50	KernelBase.dll	CreateFileW + 0x35e	C:\Windows\SysWOW64\KernelBase.dll

Рис. 17-26. Открытие файла функцией ***CreateFileW*** (фрейм 50) инициирует операции чтения файлов драйвером SRTSP64.Sys

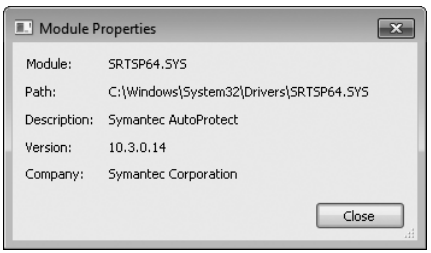


Рис. 17-27. Окно свойств модуля SRTSP64.SYS

Обычно администраторы устанавливают антивирусы на файловых серверах, чтобы клиентам не приходилось проверять файлы, к которым они обращаются, так как проверки на стороне клиента попросту были бы избыточными. Поэтому второй рекомендацией инженера было установить в клиентском антивирусе исключения для общей папки, в которой хранились профили пользователей.

Менее чем за 15 минут инженер описал проведенный анализ и рекомендуемые действия и отправил их клиенту. Журнал монитора сети был нужен просто для подтверждения того, что было обнаружено в журнале Procmon. Администратор выполнил полученные инструкции и через несколько дней сообщил, что пользователь больше не жалуется. Так при помощи Procmon и стеков потоков была решена еще одна проблема.

Сложный случай с зависанием Outlook

Этим рассказом со мной поделился мой друг Эндрю Ричардс, инженер службы поддержки Microsoft Exchange Server. Он весьма интересен, поскольку описывает применение утилиты Sysinternals, разработанной специально для служб поддержки Microsoft, и состоит из описания двух проблем.

Началось все с того, что администратор одной корпорации обратился в службу поддержки Microsoft с жалобами пользователей корпоративной сети на зависания Outlook длительностью до 15 минут. Так как это указывало на проблемы с Exchange, дело было передано в службу поддержки Exchange Server.

Специалисты из этой службы создали регистраторы данных для Performance Monitor с сотнями счетчиков, полезных при решении проблем с Exchange и собирающих данные об активности LDAP, RPC и SMTP, числе подключений Exchange, использовании памяти и процессора. Администратор должен был записывать в журнал активность сервера с 12-часовыми циклами, первый из которых начинался в 9 вечера и заканчивался в 9 утра следующего дня. Когда инженеры службы поддержки просмотрели журнал, они увидели две явных тенденции, несмотря на высокую плотность данных. Во-первых, как и ожидалось, нагрузка на сервер Exchange возрастала утром, когда пользователи приходили на работу и запускали Outlook. Во-вторых, диаграммы счетчиков показывали аномальный период между 8:05 и 8:20, который в точности соответствовал длительности задержек, на которые жаловались пользователи.

Инженеры поддержки стали изучать трассировки, полученные в этот промежуток времени, и увидели падение утилизации времени ЦП сервера Exchange, снижение значения счетчика активных подключений и значительное повышение задержки отклика, но не смогли выявить причину (рис. 17-28).

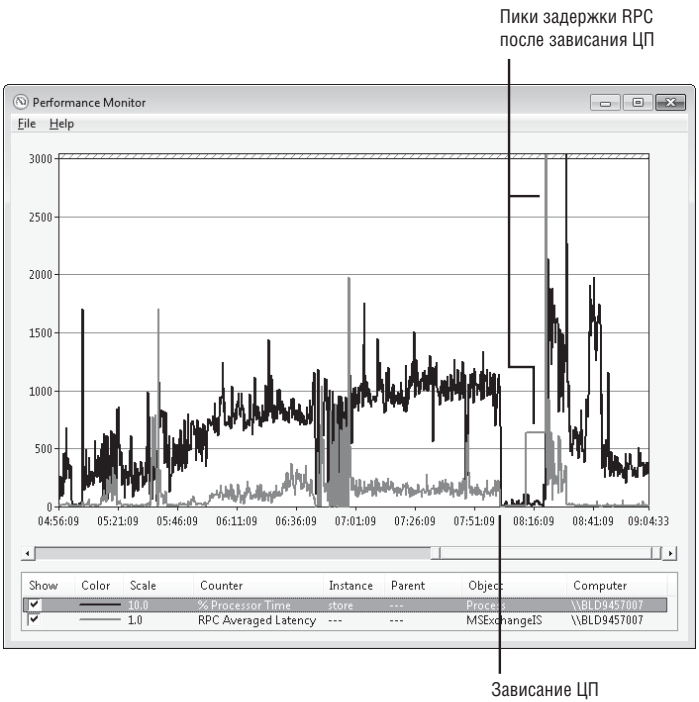


Рис. 17-28. Падение использование ЦП и увеличение задержки RPC

Они передали дело на следующий уровень службы поддержки, и за него взялся Эндрю. Эндрю изучил журналы и пришел к выводу, что требуется дополнительная информация о том, что делает Exchange во время простоя. В частности, ему нужен был дамп памяти Exchange, полученный в период, когда сервер не отвечает на запросы, включающий содержимое адресного пространства процесса, в том числе данные и код, а также состояние регистров потоков процесса. Дамп процесса Exchange позволил бы проверить потоки Exchange и узнать, что приводит к их простою.

Один из способов получить дамп — подключить к процессу отладчик, такой как Windbg из пакета средств отладки для Windows (Debugging Tools for Windows, включен в SDK Windows), и выполнить команду .dump. Однако это довольно сложно: требуется скачать и установить эти утилиты, запустить отладчик, подключить его к нужному процессу и только потом можно сохранить дампы. Вместо этого Эндрю попросил администратора скачать ProcDump. При помощи ProcDump легко получать дампы процессов, в том числе с указанным интервалом. Администратор должен был запустить ProcDump, когда производительность ЦП сервера упадет в следующий раз, чтобы сгенерировать пять дампов Store.exe, основного процесса Exchange Server, с интервалом три секунды:

```
procdump -n 5 -s 3 store.exe c:\dumps\store_mini.dmp
```

На следующий день проблема повторилась, и администратор отправил Эндрю файла дампа, полученные ProcDump. Часто процесс временно зависает из-за того, что один из его потоков устанавливает блокировку для защиты данных, которые необходимы другим потокам, и удерживает ее, выполняя какую-либо длинную операцию. Поэтому Эндрю сначала проверил блокировки. Наиболее часто при синхронизации процессов используется блокировка «критическая секция». Команда !locks debugger позволяет просмотреть в дампе заблокированные критические секции, идентификатор потока, установившего блокировку, и число ожидающих потоков. Эндрю использовал аналогичную команду !critlist расширения отладчика Sieext.dll[3]. Выходные данные показали, что несколько потоков ожидают освобождения потоком 223 критической секции:

```
0:000> !sieext.critlist
CritSec at 608e244c. Owned by thread 223.
  Waiting Threads: 43 218 219 220 221 222 224 226 227 228 230 231 232 233
```

Теперь нужно было посмотреть, что делает поток, устанавливающий блокировку. Это могло указать на код, ответственный за длительные задержки. Эндрю переключился в контекст потока-владельца, используя команду ~, и выгрузил дамп стека потока командой k:

```
0:000> ~223s
```

[3] С сайта microsoft.com можно скачать общедоступную версию — SieExtPub.dll

```
eax=61192840 ebx=00000080 ecx=0000000f edx=00000074 esi=7c829e37 edi=40100080
eip=7c82860c esp=61191c40 ebp=61191cdc icpl=0         nv up ei pl nz na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000202
ntdll!Ki FastSystemCall Ret:
7c82860c c3               ret

0:223> knL
 # ChildEBP  RetAddr
00 61191c3c  7c826e09 ntdll!KiFastSystemCall Ret
01 61191c40  77e649ff ntdll!ZwCreateFile+0xc
02 61191cdc  608c6b70 kernel 32!CreateFileW+0x377
WARNING: Stack unwind information not available. Following frames may be wrong.
03 61191cfc  7527e1a6 SAVFMSEVSAPI+0x6b70
04 00000000  00000000 0x7527e1a6
```

Как иногда бывает, отладчик не знал, как интерпретировать стек, встретив фрейм, указывающий на Savefmsevsapi — образ, для которого не смог получить символы. Символы для большинства образов Microsoft опубликованы на сервере символов Microsoft, а значит, это, скорее всего, была сторонняя библиотека DLL, загруженная в процесс Store.exe, на которую и пало подозрение. Команда lm, выводящая список модулей и информацию о версиях и путях загруженных образов, показала, что Savfmsevsapi — часть программы антивирусной защиты почты от Symantec:

```
0:000> lmvm SAVFMSEVSAPI
start     end            module name
608c0000  608e9000       SAVFMSEVSAPI T  (no symbols)
    Loaded symbol   image file:   SAVFMSEVSAPI.dll
    Image path:  C:\Program Files\Symantec\SMSMSE\6.0\Server\SAVFMSEVSAPI.dll
    Image path:  C:\Program
    Image name:  SAVFMSEVSAPI.dll
    Timestamp:        Wed Jul 08 03:09:42 2009 (4A547066)
    CheckSum:         00033066
    ImageSize:        00029000
    File version:     6.0.9.286
    Product version:  6.0.9.286
    File flags:       0 (Mask 0)
    File OS:          10001 DOS Win16
    File type:        1.0 App
    File date:        00000000.00000000
    Translations:     0000.04b0 0000.04e4 0409.04b0 0409.04e4
```

Эндрю проверил другие дампы, и нашел в них то же самое. Все указывало на виновника — программу от Symantec. Эндрю, с разрешения администратора, отправил дампы и результаты анализа в службу поддержки Symantec. Через несколько часов оттуда сообщили, что дампы действительно указывают на проблему с последней версией антивируса, и прислали исправление ошибки. Администратор установил его и продолжил мониторинг, чтобы

убедиться в устранении проблемы. Производительность вернулась на обычный уровень, а длинные задержки исчезли.

Однако через какое-то время администратор стал получать жалобы от некоторых пользователей Outlook на единичные случаи зависания длительностью до минуты. Эндрю попросил прислать соответствующий 12-часовой журнал Performance Monitor, но в этот раз очевидной аномалии не было.

Предполагая, что зависания будут видны в истории использования ЦП процессом Store.exe, он удалил все счетчики, кроме счетчика ЦП, и обнаружил три скачка в утренние часы, когда пользователи начинали входить в систему и повышать нагрузку на сервер (рис. 17-29).

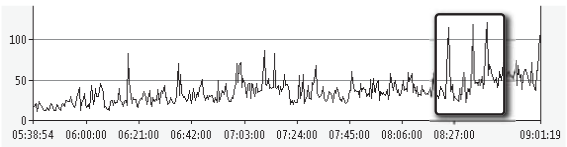


Рис. 17-29. Скачки использования ЦП процессом Store.exe примерно в 8:30 утра

Поскольку сервер был восьмиядерным, диапазон значений счетчиков утилизации ЦП процессом был 0–800, так что скачки были далеки от того, чтобы замедлить систему, но они были явно выше обычного. При дальнейшем рассмотрении и масштабирования вертикальной шкалы для выделения скачков Эндрю обнаружил, что средняя загруженность ЦП всегда была ниже 75% одного ядра, а пики длились 15-30 секунд (рис. 17-30).

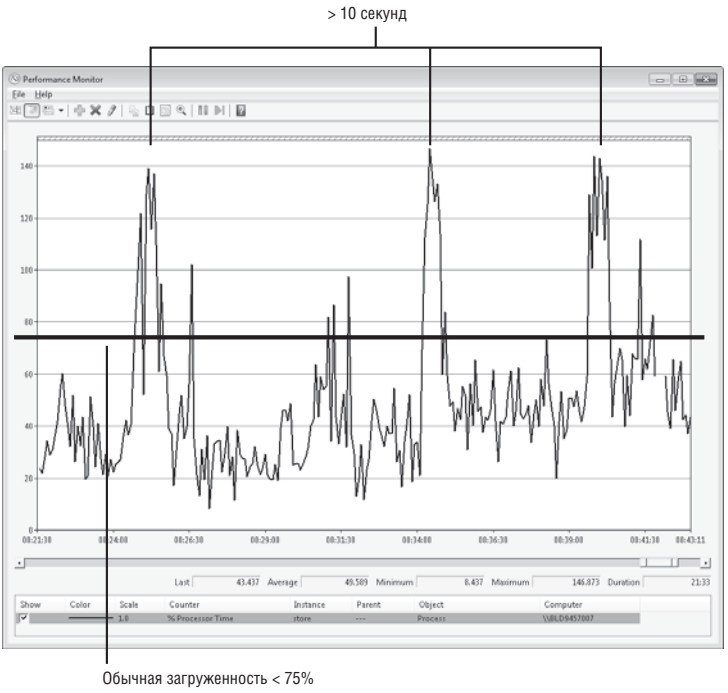


Рис. 17-30. Просмотр пиков использования ЦП

Что делал сервер Exchange во время пиков? Они были слишком короткими и случайными по амплитуде, чтобы администратор смог запустить ProcDump и получить необходимые дампы. К счастью, я разработал ProcDump с учетом именно такой ситуации и предусмотрел поддержку нескольких условий создания дампа. Например, можно настроить ProcDump для генерации дампа процесса, когда тот завершается, или когда его закрытая память превышает определенный объем, и даже когда достигается указанное значение счетчика производительности. Однако основное условие запуска — превышение процессом указанного порога утилизации ЦП в течение указанного промежутка времени.

Из журнала Performance Monitor Эндрю получил информацию, необходимую для составления условия получения дампов при возникновении пиков использования ЦП:

```
procdump.exe -n 20 -s 10 -c 75 -u store.exe c:\dumps\store_75pc_10sec.dmp
```

С такими параметрами ProcDump генерирует дамп процесса Store.exe, когда использование ЦП этим процессом превышает 75% (-c 75) для одного ядра (-u) в течение 10 секунд (-s 10). Перед завершением работы генерируется 20 дампов (-n 20) в папке C:\Dumps под именами, начинающимися с «store_75pc_10sec». Администратор выполнил эту команду, перед уходом домой, а на следующее утро имел 20 дампов. Он отправил их Эндрю, который просмотрел их в отладчике Windbg по очереди.

Когда Procdump генерирует дамп по условию о превышении использования ЦП, в файле дампа устанавливается контекст потока, потреблявшего наибольшее количество ресурсов ЦП во время создания дампа. Поскольку команды отладчика для выгрузки дампа стека выполняются в контексте текущего потока, достаточно просто выполнить их для просмотра стека процесса, с наибольшей вероятностью вызвавшего пик утилизации ЦП. Больше половины дампов не содержали полезной информации, так как были получены уже после скачка, или когда выполнялся код, явно не имеющий отношения к проблеме. Однако в нескольких дампах были данные, аналогичные показанным на рис. 17-31.

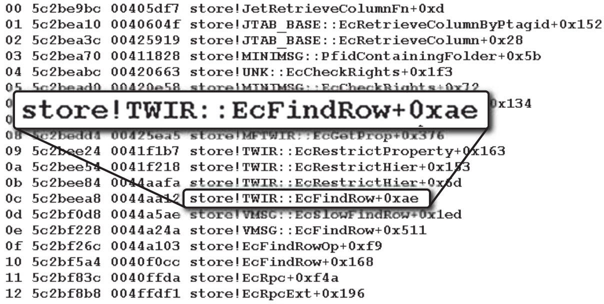


Рис. 17-31. Стек Store.exe с инструкцией ***store!TWIR::EcFindRow+0xae***

Выделенный на рисунке фрейм указывает на функцию Store *EcFindRow*, что говорит о том, что пики вызваны длительными запросами к базе данных. Такого рода запросы выполняются, когда Outlook обращается к папке почтового ящика, содержащей тысячи элементов. Эндрю предложил администратору проверить большие почтовые ящики и дал ссылку на статью службы поддержки Exchange, в которой рассказывается, как делать это в каждой из версий Exchange (*http://msexchangeteam.com/archive/2009/12/07/453450.aspx*).

Созданный сценарий показал, что у нескольких пользователей в папках хранятся десятки тысяч писем. Администратор попросил этих пользователей оставить не более 5000 писем (это рекомендованное число для Exchange 2003, в каждой версии оно увеличивается, и для Exchange 2010 составляет 100000), заархивировав их, удалив или переложив в другие папки. Через пару дней пользователи привели свои почтовые ящики в порядок и жалобы полностью исчезли, а последующий мониторинг показал, что проблема устранена.

Так ProcDump помог распутать сложный случай с зависанием Outlook.

Глава 18

Вредоносные программы

Вредоносные программы являются причиной многих компьютерных проблем. По определению они всегда выполняют действия, нежелательные для пользователя. Иногда они пытаются сделать это по-тихому, чтобы пользователь не заметил их, а иногда — явно, как описано в разделе «Дело о вредоносном ПО, завершающем процессы» этой главы. Как и многие обычные программы, вредоносные программы нередко содержат ошибки в коде. Но, в отличие от большинства нормальных программ, вредоносные часто *активно* препятствуют их обнаружению и удалению.

- **Случай с вирусом, блокирующим утилиты Sysinternals**, интересен, потому что касается программы, целенаправленно блокирующей запуск утилит Sysinternals. Естественно, проблема была решена с помощью утилит Sysinternals.
- **Случай с вирусом, завершающим процессы** произошел, когда мы заканчивали писать эту книгу. Знакомый Аарона принес лэптоп своего сына, чтобы почистить его. Вирус не сдавался по-хорошему, Autoruns в безопасном режиме было явно недостаточно.
- **Случай с поддельным компонентом системы** демонстрирует использование утилиты Strings для поиска вредоносного ПО.
- **Случай с таинственной точкой ASEP** описывает вирус, создающий собственную точку расширения автозапуска (Auto-start Extensibility Point, ASEP). Проблема решена при помощи ListDLLs, Procmon, Procexp и Autoruns.

Вирус, блокирующий утилиты Sysinternals

Знакомый попросил одного пользователя утилит Sysinternals взглянуть на систему, подозревая, что она заражена. Запуск и вход в систему происходили очень долго, а поиск с помощью Microsoft Security Essentials никогда не доходил до конца. Пользователь поискал необычные процессы в Диспетчере задач, но ничего особого не нашел.

Затем он обратился к утилитам Sysinternals, попробовав запустить AutoRuns, Procmon, Procexp и RootkitRevealer[1], но все утилиты закрывались сразу же после запуска. В качестве эксперимента, он попробовал открыть текстовый файл с именем «Process Explorer» в Блокноте, и он тоже сразу же закрылся. К этому моменту он заметил множество признаков заражения системы, но не знал, как найти вредоносную программу, не говоря уже об ее удалении.

Просматривая пакет Sysinternals, пользователь обнаружил утилиту Desktops. Эксперименты с Блокнотом показали, что вредоносное ПО следило за заголовками окон опасных для него программ. Поскольку при перечислении окон обрабатываются только окна, находящиеся на том же рабочем столе, что и вызывающий процесс, он предположил, что автор вредоносной программы не учел возможность запуска программ на нестандартных рабочих столах. После запуска Desktops и переключения на второй рабочий стол он смог запустить Procmon и другие утилиты (рис. 18-1, см. также главу 2).

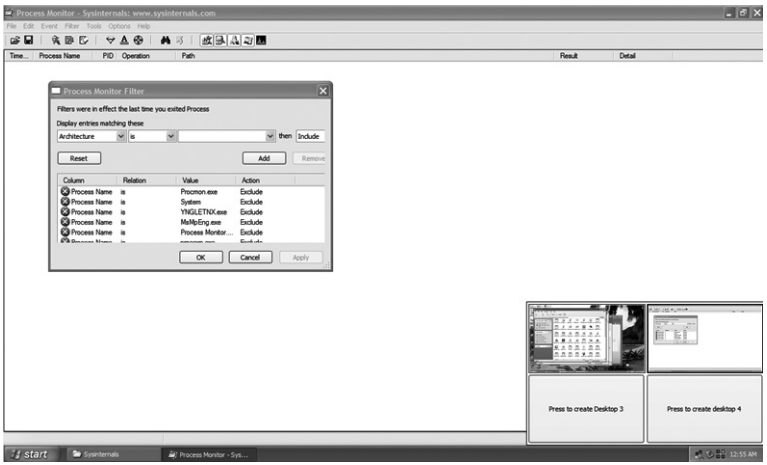


Рис. 18-1. Запуск утилит Sysinternals на другом рабочем столе

Сначала он воспользовался утилитой Procexp. Все параметры процессов выглядели как обычно, так что он включил параметры **Verify Signers** и **Verified Signer**. Оказалось, что все основные исполняемые файлы действительно те, за кого себя выдают.

Затем он запустил Procmon и заметил слишком активную работу процесса Winlogon. Он установил фильтр, чтобы наблюдать только за Winlogon (рис. 18-2), и увидел, что этот процесс каждую секунду проверяет странный раздел реестра:

```
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\ Notify\
acdcacaeaacbafbeaa
```

[1] RootkitRevealer — утилита для обнаружения руткитов, созданная мной несколько лет назад, когда руткиты были малоизвестны, и основные производители антивирусов еще не взялись за их обнаружение и удаление. С тех пор RootkitRevealer устарел и исключен из пакета Sysinternals.

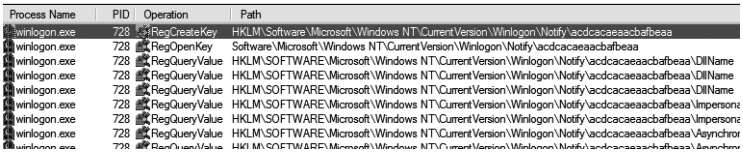


Рис. 18-2. Просмотр необычной активности Winlogon.exe в Procmon

Затем он запустил Autoruns, включив проверку подписей образов и скрыв элементы Microsoft и Windows. Имея список неподписанных программ от сторонних производителей, он быстро нашел виновника проблем: неподписанную библиотеку DLL с именем из случайного набора символов, зарегистрированную как пакет уведомлений Winlogon, загружающий в процесс Winlogon другую DLL (рис. 18-3). Он удалил эту запись в Autoruns, но обнаружил при повторной проверке, что она снова появилась.

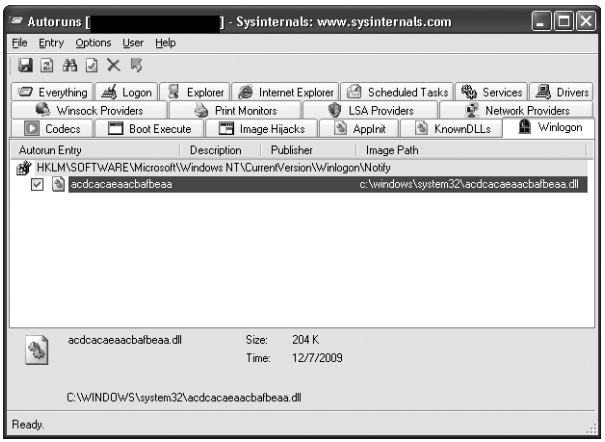


Рис. 18-3. Обнаружение с помощью Autoruns вредоносной программы, зарегистрированной как пакет уведомлений Winlogon

После этого он вернулся в Microsoft Security Essentials и запустил проверку только что найденной DLL (рис. 18-4). После ее очистки удалось удалить и запись в Autoruns, что вернуло систему в нормальное состояние.

Рис. 18-4. Удаление угрозы, обнаруженной с помощью утилит Sysinternals

Вирус, завершающий процессы

Пол, знакомый Аарона, позвонил и сказал, что на лэптопе его сына появился вирус, сообщающий, что компьютер заражен, и требующий заплатить «отступные» с банковской карты. Аарон предположил, что это могут быть «останки» какой-нибудь сомнительной веб-страницы, которые должны исчезнуть после выхода из системы. Когда перебрали все простые решения, оставалось только привезти «пациента» к Аарону

Когда Пол запустил лэптоп и ввел пароль сына, на весь экран развернулось окно. В нем программа, представившаяся как антивирус, показывала список вредоносного ПО, которым якобы заражена система, и требовала ввести реквизиты платежа по карте, чтобы удалить найденные «вирусы». Естественно, эта программа не была настоящим, купленным Полом антивирусом (который, кстати, проморгал запуск этой вредоносной программы).

Аарон достал компакт-диск с утилитами Sysinternals и попытался запустить Procexp, Autoruns и прочие утилиты. Ни одна из них не запустилась. Вспомнив предыдущий случай, он попробовал запустить Desktops, но и это не удалось. Вредоносная программа не давала запустить ни один новый процесс, включая командную строку, PowerShell и Диспетчер задач. В лучшем случае появлялась рамка окна, которая тут же исчезала.

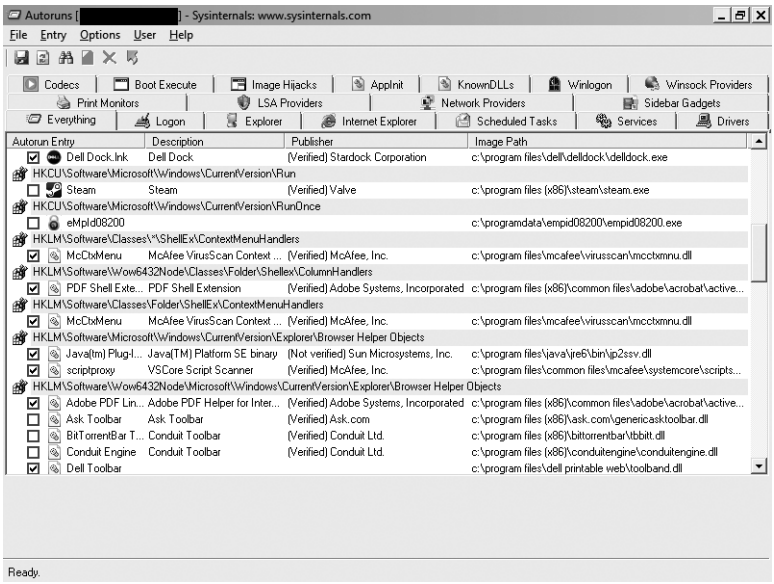


Рис. 18-5. Отключение подозрительных элементов в Autoruns в безопасном режиме

Аарон перезагрузил компьютер в безопасном режиме с командной строкой, в котором загружается минимальный набор драйверов, а вместо Проводника Windows запускается Cmd.exe. Кроме того, в этом режиме обрабатывается очень мало точек ASEP (см. гл. 5). Вредоносное ПО не запустилось, что говорило о том, что оно запускалось из одной из пропущенных

точек ASEP. Аарон запустил Autoruns, включив проверку подписей и скрыв элементы Windows и Microsoft. Обнаружилось несколько подозрительных элементов, включая несколько программ, открывающих общий доступ к файлам, странные панели инструментов Internet Explorer и вспомогательные объекты обозревателя. Все эти элементы он отключил, а не удалил (рис. 18-5), на случай, если дело не в них: их папки были созданы довольно давно, следовательно, они вряд ли являлись причиной проблемы.

Вирус удалось обнаружить довольно легко: это была программа с именем eMpld08200, без описания и издателя, запускавшаяся разделом HKCU RunOnce из папки C:\ProgramData и, в дополнение ко всему, напялившая на себя значок настоящего антивируса. Аарон удалил эту точку ASEP в Autoruns и соответствующие файлы через командную строку (рис. 18-6). На всякий случай, он оставил отключенными ненужные программы и расширения Internet Explorer. После перезагрузки компьютер заработал без проблем.

Интересно, что этот вирус не имел администраторских прав, установился в папку, доступную для записи пользователям, и постоянно запускался через пользовательскую, а не глобальную ASEP. Через несколько недель такой же вирус заразил Windows XP на компьютере тещи Аарона. Так как он позаботился, чтобы теща работала только под учетной записью обычного пользователя, он легко смог удалить вирус, войдя под администраторской учетной записью, под которой вирус не запускался. Аарон запустил Autoruns, выбрал в меню **User** инфицированную учетную запись и удалил проблемную точку ASEP (к сожалению, снимков экрана он не сделал). Отсюда можно сделать два вывода: у вирусов становится все больше возможностей для причинения ущерба и без администраторских прав; такие вирусы гораздо легче удалять, чем вредоносные программы, нарушающие целостность операционной системы.

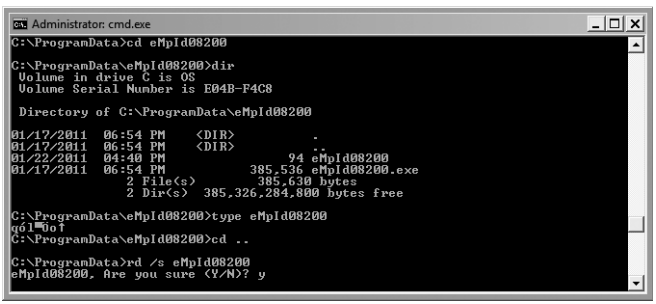


Рис. 18-6. Удаление вредоносного ПО из командной строки в безопасном режиме

Дело о поддельном компоненте системы

Следующие два случая описаны Грегом Коттингемом, старшим инженером службы поддержки Microsoft. В сентябре 2010 г. команда Грега начала получать сразу от нескольких компаний сообщения о новом черве, который затем получил имя Win32/Visal.b. Грегу было поручено разобраться, и он на-

чал с нажатия Ctrl+Shift+Esc, чтобы запустить Диспетчер задач. На первый взгляд, ни один из показанных процессов (рис. 18-7) не вызывал подозрений у нетренированного наблюдателя. Но если снять флажок **Show Processes From All Users (Отображать процессы всех пользователей)**, в списке должен быть только один процесс Csrss.exe, а их было два. (В действительности, этот флажок определяет, отображаются ли процессы только из текущего или из всех *сеансов служб терминалов*, подробнее см. в главе 2.)

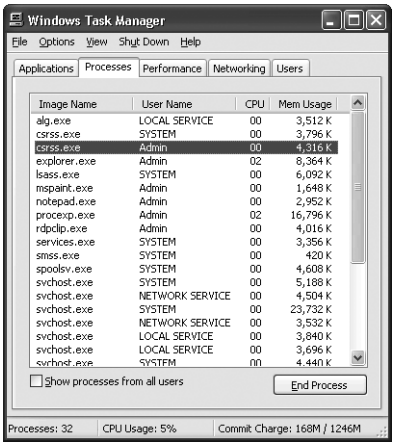


Рис. 18-7. Диспетчер задач показывает два экземпляра Csrss.exe в одном сеансе

Одно из ограничений Диспетчера задач в том, что он не показывает полный путь к исполняемым образам. Вредоносное ПО часто прячется за именами обычных программ, такими как Svchost.exe и Csrss.exe, но устанавливается не в %windir%\System32, где находятся настоящие файлы, а в другие папки, например, в %windir%. Procexp не имеет этого недостатка и показывает полный путь к исполняемому файлу во всплывающей подсказке (рис. 18-8) или в отдельном столбце.

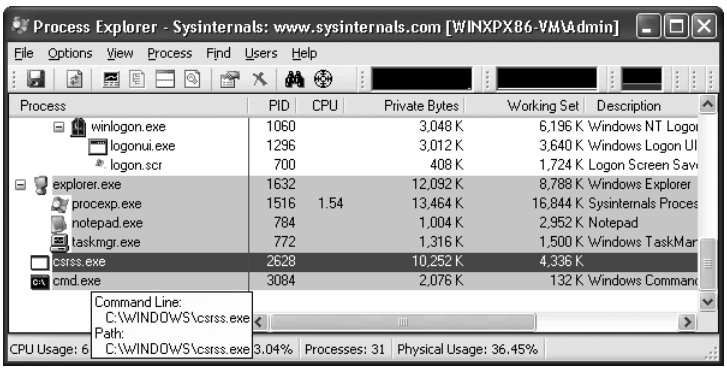


Рис. 18-8. Отображение пути к «лишнему» процессу Csrss.exe в Procexp

Определив, что «лишний» экземпляр Csrss.exe находился в %windir% и не прошел проверку подписи, Грег запустил утилиту Strings, чтобы понять, что это за файл (рис. 18-9). Проверка выявила несколько признаков вредоносной программы, включая код для создания файла Autorun.inf, копирующегося на съемные диски и запускающего вирус при вставке диска с этим Autorun.inf в другой компьютер, а также перечисление компьютеров и общих папок и копирование вируса в эти папки с обманчивыми именами и расширениями.

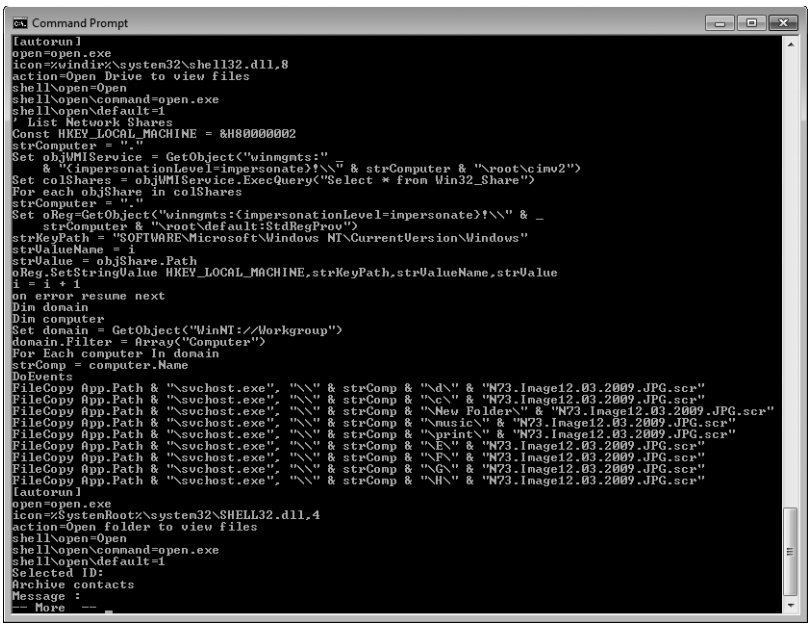


Рис. 18-9. Выявление вредоносного ПО в поддельном файле Csrss.exe при помощи Strings

С помощью Stings Грег также обнаруживал файлы вредоносных программ по сигнатуре «UPX0» (свидетельствующей о сжатии файла) и «кустарным» путям к PDB-файлам символов вроде «d:\hack.86» и «c:\mystuff».

Убедившись, что этот поддельный компонент Windows действительно был вредоносной программой, Грег и его команда стали работать с Центром защиты от вредоносного ПО, чтобы документировать поведение этого вируса и выработать решение для защиты от него.

Случай с таинственной точкой ASEP

Грегу поручили дело клиента, представляющего крупную сеть больниц в США, в котором сообщалось об атаке и заражении вирусом Marioforever. Клиент обнаружил вирус, когда его принтеры стали печатать гигантские объемы случайного текста, что приводило к замедлению сети и большому расходу бумаги. Антивирусное ПО определило файл Marioforever.exe в папке %Systemroot% одного из компьютеров, как подозрительный, но после

удаления при следующей перезагрузке этот файл появлялся снова. Другие антивирусные программы вообще не обращали на этот файл внимания.

Грег начал с поиска других подозрительных файлов в папке %SystemRoot% одной из инфицированных систем и нашел недавно созданный файл Nvrsma.dll. Это имя носит один из компонентов видеодрайвера Nvidia, на том компьютере не было устройств от Nvidia. Когда Грег попытался удалить и переименовать этот файл, он получил странную ошибку нарушения совместного доступа. Это означало, что какой-то процесс открыл этот файл и не давал открыть его другим процессам. Получить список процессов, открывших файл или загрузивших DLL, можно с помощью нескольких утилит Sysinternals, включая Process Explorer и Handle. Так как подозрительный файл был библиотекой DLL, Грег решил использовать утилиту Listdlls, которая показала, что эта библиотека загружена только процессом Winlogon:

```
C:\>listdlls -d nvrsma.dll

ListDLLs v2.25 - DLL lister for Win9x/NT
Copyright (C) 1997-2004 Mark Russinovich
Sysinternals - www.sysinternals.com

------------------------------------------------------------------------
winlogon.exe pid:   416
Command line:  winlogon.exe

Base          Size      Version     Path
0x10000000    0x34000               C:\WINDOWS\system32\nvrsma.dll
```

Winlogon — основной системный процесс, отвечающий за управление интерактивными сеансами входа. В данном случае он и был хостом для вредоносной библиотеки DLL. Следующим этапом было определение способа загрузки DLL в Winlogon. Загрузка должна была выполняться через точку автозапуска, поэтому Грег запустил Autoruns и AutorunsC (консольный вариант). Однако никаких признаков Nvrsma.dll не было, а все элементы автозапуска были либо компонентами Windows, либо другими легальными компонентами. Это был тупик, поэтому Грег вернулся к Procmon.

Winlogon запускается в ходе загрузки, поэтому Грег включил в Procmon функцию записи информации о загрузке, перезагрузил систему, запустил Procmon и загрузил полученный журнал. Затем он нажал Ctrl+F и выполнил поиск текста «nvrsma». На рис. 18-10 показано, что он нашел: первая ссылка встречалась, когда Winlogon.exe запрашивал значение раздела реестра HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Windows\dzplnit_DLLs, которое возвращало текстовое значение «nvrsma». Несколько событий спустя Winlogon.exe открывал и проецировал nvrsma.dll в память.

После этого Грег просмотрел стек вызовов для этого первого события реестра. Как видно на рис. 18-11, чтение реестра инициировано библиотекой User32.dll. Грег знал, что имя «dzpInit_DLLs» очень похоже на хорошо известную и ши-

роко используемую вредоносными программами точку ASEP «AppInit_DLLs», определенную в том же разделе реестра и также инициируемую из User32.dll[2]. Но эта библиотека была не оттуда. Была ли dzpInit_DLLs новой точкой ASEP, о которой ни Грег, ни автор Autoruns никогда не слышали?

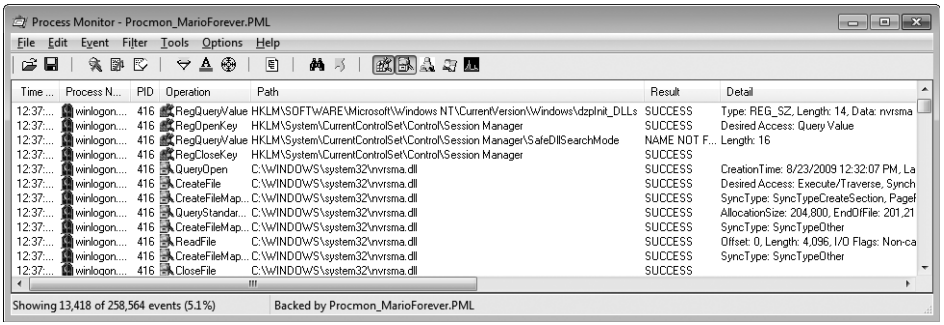


Рис. 18-10. Procmon показывает, почему Winlogon.exe загрузил nvrsma.dll

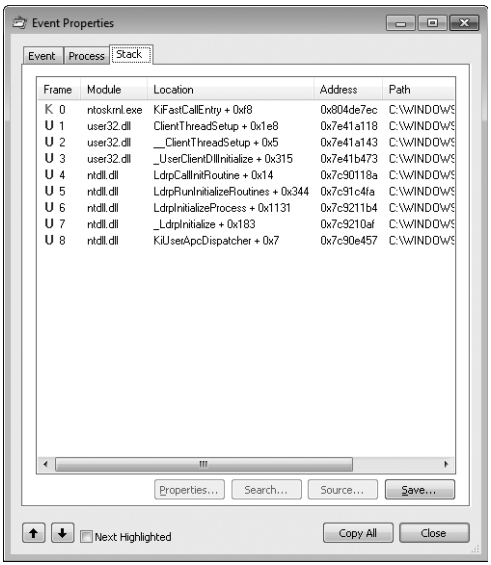


Рис. 18-11. Стек вызовов, показывающий событие реестра, инициированное User32.dll

Теперь Грег обратил внимание на User32.dll. Он заметил, что на инфицированных системах дата последнего изменения User32.dll в папках System32 и DllCache совпадала с датой первого заражения. Внимательнее просмотрев данные в Autoruns, он увидел, что подпись библиотеки User32.dll не прошла проверку (рис. 18-12), а значит, библиотека изменена или полностью заменена.

---

[2] Когда процесс в Windows XP и предыдущих версиях ОС запускает User32.dll, он также загружает все библиотеки DLL, указанные в значении реестра AppInit_DLLs. В Autoruns эти DLL показаны на вкладке AppInit.

Грег запустил Procexp в нормально работающей системе Windows XP и в инфицированной. В обеих он выбрал процесс Winlogon.exe, открыл представление DLL, дважды щелкнул на имени User32.dll в нижней части окна, чтобы открыть диалоговое окно свойств, перешел на вкладку **Strings** и сравнил текст, показанный в двух системах. Совпадало все, кроме одного: на инфицированной системе точка запуска AppInit_DLLs была заменена точкой dzpInit_DLLs (рис. 18-13). Сравнив двоичный код нормального и инфицированного файлов User32.dll при помощи команды Windows `fc /b`, Грег обнаружил, что эти два байта были единственным различием между двумя файлами. Вредоносная программа создала собственную точку ASEP, изменив два байта в User32.dll, чтобы загружать DLL, заданные параметром реестра dzpInit_DLLs, а не AppInit_DLLs.

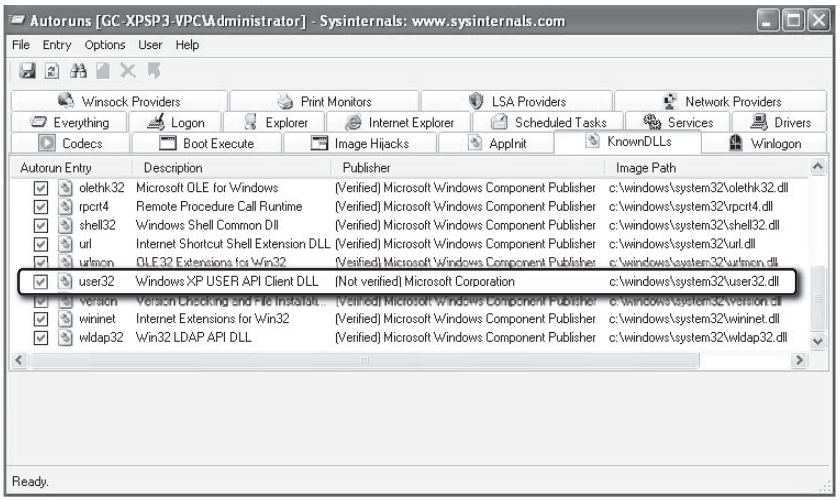


Рис. 18-12. Библиотека User32 не прошла проверку подписей в Autoruns

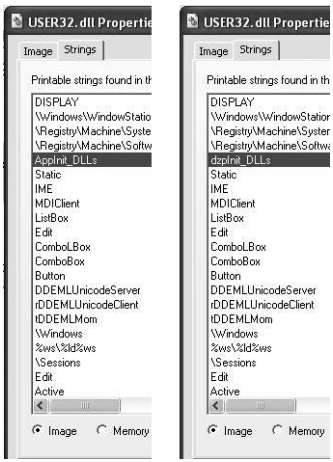


Рис. 18-13. Сравнение текстовых строк в нормальном файле User32.dll (слева) и инфицированном (справа)

Зная, как именно активируется основная DLL вредоносной программы, Грег смог очистить систему. Поскольку библиотека User32.dll всегда блокируется вредоносной программой во время работы системы, он загрузил среду WinPE с компакт-диска и скопировал оттуда нормальный файл User32.dll, переписав инфицированный. Затем он удалил остальные вредоносные файлы, которые обнаружил во время поисков. После этого он перезагрузил систему и убедился, что она очищена. Дело было закрыто, администраторы больничной сети получили инструкции по удалению вредоносного ПО, а антивирусная группа Microsoft — материал для добавления в Forefront и Malicious Software Removal Toolkit. Так Грег распутал загадочное на первый взгляд дело, используя несколько утилит Sysinternals, и помог восстановить нормальную работу больниц.

Предметный указатель

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

Г

Д

Ж

З

И

К

Л

М

Н

О

П

P

С

Т

У

Ф

Х

Ц

Ч

Э

Я

Об авторах

Марк Руссинович — технический специалист, в команде Microsoft Windows Azure работает над ОС от Microsoft, предназначенной для центров обработки данных. Он признанный эксперт по внутреннему устройству Windows, а также по обеспечению безопасности и проектированию операционных систем. Марк — автор недавно опубликованного кибертриллера «Zero Day» и соавтор множества книг, вышедших в издательстве Microsoft Press. Он пришел в Microsoft в 2006 г., когда корпорация выкупила фирму Winternals Software, основанную им в 1996 г., вместе с сайтом Sysinternals, на котором Марк публиковал десятки созданных им популярных программ для администрирования и диагностики Windows. Без Марка не обходится ни одна серьезная ИТ-конференция, включая Microsoft TechEd, WinHEC и Professional Developers Conference.

С Марком можно связаться по адресу *markruss@microsoft.com*. Следите за его новостями в Твиттере: *http://www.twitter.com/markrussinovich*.

Аарон Маргозис — ведущий консультант служб взаимодействия с клиентами Microsoft Public Sector, где с 1999 г. он работает, в основном, с клиентами из федерального правительства США. Аарон специализируется на разработке приложений для платформ Microsoft с упором на безопасность и совместимость. Он — авторитетный докладчик на конференциях Microsoft, а также известный сторонник работы в Windows XP без административных привилегий, а также автор утилит и руководств по этой теме. Его сценарий MakeMeAdmin стал первым примером кода, позволяющего одному и тому же пользователю работать как обычному пользователю и администратору одновременно. Эта новинка оказала большое влияние на разработку механизма UAC. Некоторые утилиты, созданные Аароном, можно скачать через его блог (*http://blogs.msdn.com/aaron_margosis*) или блог его команды (*http://blogs.technet.com/fdcc*).

Связаться с Аароном можно по адресу *aaronmar@microsoft.com*.

Утилиты Sysinternals

Справочник администратора

Независимо от вашего опыта работы с утилитами Sysinternals и систем, которыми вы управляете, вы обнаружите новые возможности, получите советы и узнаете о приемах, которые позволят эффективнее решать сложнейшие проблемы с Windows.

Весь пакет утилит, содержащий более 70 мощных средств для диагностики и устранения сбоев Windows-систем, доступен для бесплатной загрузки на сайте Microsoft TechNet по адресу: **www.sysinternals.com**.

В этой книге:

- диагностика перегрузки ЦП, утечек памяти и прочих сбоев;
- получение полной картины активности файловой системы, дисков, реестра, процессов, потоков и сети;
- диагностика и устранение неполадок Active Directory®;
- поиск, просмотр и отключение автозапуска приложений и компонентов;
- мониторинг отладочного вывода приложений;
- автоматическая генерация дампов памяти для поиска причин сбоев программ;
- проверка и анализ цифровых подписей файлов и других данных, связанных с ИТ-безопасностью;
- работа Sysinternals на удаленных компьютерах;
- подробные инструкции по Process Explorer, Process Monitor и Autoruns.

Каждая книга из серии **Справочник администратора** объединяет в себе руководство по эксплуатации и подробный справочник по основным функциям и параметрам системы.

Windows

ISBN 978-5-7502-0411-3

9 785750 204113

Издательство
БХВ-Петербург
Санкт-Петербург,
Измайловский пр., 29
E-mail: mail@bhv.ru
Internet: www.bhv.ru
Тел.: (812) 251-4244

Издательство
Русская редакция
Москва,
3-я Хорошевская ул., 11
E-mail: info@rusedit.com
Internet: www.rusedit.com
Тел.: (495) 638-5-638

bhv

РУССКАЯ РЕДАКЦИЯ