Оглавление

Моей семье — жене Идит и нашим детям Даниэль, Амиту и Йоаву;
спасибо вам за терпение и поддержку во время этой непростой работы.

Павел Йосифович

Моим родителям, которые направляли и вдохновляли меня на моем пути
к мечте, и моей семье, которая была рядом со мной всеми этими
бесчисленными вечерами.

Алекс Ионеску

Нашим родителям, которые направляли и вдохновляли нас
на пути к мечте.

Марк Руссинович и *Дэвид Соломон*

Введение

Седьмое издание книги *«Внутреннее устройство Windows»* предназначено для профессионалов (разработчиков, специалистов по безопасности и системных администраторов), желающих более глубоко разобраться в работе основных компонентов Microsoft Windows 10 и Windows Server 2016. Разработчики смогут лучше понять обоснование того или иного проектного решения при создании приложений для Windows, и смогут успешнее производить отладку сложных проблем.

Книга пригодится и системным администраторам, так как понимание «скрытых» принципов работы операционной системы позволяет лучше понять ее поведение и облегчить устранение неполадок в системе, если возникают сбои.

Специалисты по безопасности узнают, как злоумышленники могут воспользоваться уязвимостями операционной системы и вызвать нежелательное поведение, а также ознакомятся с защитными мерами и средствами безопасности, реализованными в современных версиях Windows.

Прочитав эту книгу, вы будете лучше разбираться в работе Windows и в причинах того или иного поведения ОС.

История книги

Это седьмое издание книги, которая сначала называлась «*Inside Windows NT*» (Microsoft Press, 1992) и была написана Хелен Кастер (Helen Custer) еще до выхода Microsoft Windows NT 3.1. «*Inside Windows NT*» была первой книгой, написанной о Windows NT и предоставившей ключевую информацию о сути архитектуры и конструкции системы. «*Inside Windows NT, Second Edition*» (Microsoft Press, 1998) была написана Дэвидом Соломоном (David Solomon). Она дополнила исходную книгу описанием Windows NT 4.0, а материал излагался на более глубоком техническом уровне.

Книга *«Inside Windows 2000, Third Edition»* (Microsoft Press, 2000) вышла под авторством Дэвида Соломона (David Solomon) и Марка Руссиновича (Mark Russinovich). В нее было добавлено множество новых тем, например: запуск и завершение работы, внутреннее устройство служб, реестра, драйверов файловой системы и сети. В ней также были рассмотрены изменения, внесенные в ядро Windows 2000, на-

пример: модель драйверов Windows Driver Model (WDM), Plug and Play, диспетчер энергопотребления, Windows Management Instrumentation (WMI), шифрование, объект задания и службы терминалов. В книгу «*Windows Internals, Fourth Edition*» были включены обновления, связанные с выходом Windows XP и Windows Server 2003, и дополнительный контент, предназначенный для IT-профессионалов в применении их знаний внутреннего устройства Windows — например, в использовании основных инструментов из комплекта Windows Sysinternals и в анализе аварийных дампов.

Книга «*Windows Internals, Fifth Edition*» (Microsoft Press, 2009) была обновлена под выход Windows Vista и Windows Server 2008. В это время Марк Руссинович перешел на полную ставку в Microsoft (где он теперь является техническим директором Azure), а у книги появился новый соавтор Алекс Ионеску. В новом материале описан загрузчик образов, средства отладки пользовательского режима, механизм ALPC (Advanced Local Procedure Call) и Hyper-V. Следующее издание, «*Windows Internals, Sixth Edition*»[1] (Microsoft Press, 2012), было полностью обновлено с учетом многочисленных изменений ядра в Windows 7 и Windows Server 2008 R2, с добавлением множества новых экспериментов в соответствии с изменениями в инструментарии.

Изменения, внесенные в седьмое издание

С выхода последнего издания этой книги система Windows прошла несколько обновлений, конечным результатом которых стал выход Windows 10 и Windows Server 2016. Система Windows 10, которая в настоящее время считается основной версией Windows, прошла несколько изданий, от первого выпуска до производственной версии. Каждая версия помечается номером версии из четырех цифр, обозначающим год и месяц выпуска, — например, Windows 10, версия 1703, которая была опубликована в марте 2017 года. Отсюда следует, что система Windows с выхода Windows 7 прошла как минимум шесть версий (на момент написания книги).

Начиная с Windows 8, корпорация Microsoft запустила процесс конвергенции ОС, полезный с точки зрения как разработчика, так и команды разработки Windows. В Windows 8 и Windows Phone 8 все началось с конвергенции ядра, затем в Windows 8.1 и Windows Phone 8.1 процесс продолжился конвергенцией современных приложений. История конвергенции завершилась в системе Windows 10, работающей на настольных/портативных компьютерах, серверах, XBOX One, телефонах (Windows Mobile 10), HoloLens и различных IoT-устройствах (Internet of Things).

С завершением грандиозной унификации пришло время для нового издания серии, которое наконец-то синхронизировалось с почти пятилетними изменениями и по-

[1] Внутреннее устройство Microsoft Windows. 6-е изд. — СПб.: Питер, 2013. — 800 с.: ил. — (Серия «Мастер-класс»).

явлением более стабильной архитектуры ядра. Соответственно, в новом издании книги рассматриваются аспекты Windows с Windows 8 до Windows 10, версия 1703. Кроме того, в новом издании в число соавторов вошел Павел Йосифович.

Практические эксперименты

Даже без доступа к исходному коду Windows вы многое можете почерпнуть о внутреннем устройстве Windows из таких инструментальных средств, как отладчик ядра, утилиты из пакета Sysinternals и инструменты, разработанные специально для этой книги. Когда инструментальное средство может быть использовано для показа или демонстрации некоторых аспектов внутреннего поведения Windows, действия, которые можно попытаться выполнить самостоятельно, описаны во врезках «ЭКСПЕРИМЕНТ». Они встречаются по всей книге, и мы хотим, чтобы вы провели все эти эксперименты по мере чтения книги — наглядное доказательство внутренней работы Windows сильнее отпечатается в вашей памяти, чем простое чтение описания этой работы.

Незатронутые темы

Windows это большая и сложная ОС. В этой книге мы не описываем все, что имеет отношение к внутреннему устройству Windows, но делаем упор на рассмотрение основных системных компонентов. Например, в книге не дается описание COM+, распределенной объектно-ориентированной программной инфраструктуры Windows, или среды Microsoft .NET Framework, которая является основой приложений с управляемым кодом. Поскольку книга посвящена внутреннему устройству, а не использованию, программированию или системному администрированию, в ней не рассматриваются вопросы использования, программирования или настройки Windows.

Предупреждение и предостережение

Поскольку в данной книге рассматривается недокументированное поведение внутренней архитектуры и функционирования операционной системы Windows (например, внутренних структур и функций ядра), в материале книги возможны изменения между выпусками.

Под «возможными изменениями» не обязательно имеется в виду, что подробности, рассмотренные в данной книге, будут меняться от выпуска к выпуску, но не рассчитывайте на то, что они не претерпят вообще никаких изменений. Любое программное обеспечение, использующее эти недокументированные интерфейсы в будущих релизах Windows, может оказаться неработоспособным. Хуже того, программы,

работающие в режиме ядра (например, драйверы устройств) и использующие эти недокументированные интерфейсы, могут при запуске новых выпусков Windows вызвать фатальный сбой с возможной потерей данных.

Одним словом, никогда не используйте внутреннюю функциональность Windows, разделы реестра, поведение, API или любые другие недокументированные подробности, описанные в книге, при разработке любых программ, предназначенных для конечного пользователя или любых других целей, кроме исследования и документирования. Поиск официальной документации по конкретным темам всегда следует начинать с MSDN (Microsoft Software Development Network).

Что мы ожидаем от читателя

Книга предполагает, что читатель уверенно работает с Windows на уровне опытного пользователя, а также понимает основные концепции операционных систем и оборудования: регистры процессора, память, процессы и программные потоки. В некоторых разделах книги также может пригодиться понимание функций, указателей и других конструкций языка программирования C.

Структура книги

Книга разделена на две части (как и в шестом издании); первую часть вы сейчас держите в руках.

- Глава 1 «Концепции и средства» знакомит читателя с концепциями внутреннего строения Windows и представляет основные инструменты, используемые в книге. Чрезвычайно важно начать чтение с этой главы, потому что в ней содержится вся вводная информация, необходимая для понимания остального материала.
- В главе 2 «Архитектура системы» представлена архитектура и основные компоненты Windows; некоторые из них изложены достаточно подробно. Другие концепции более подробно рассматриваются в последующих главах.
- Глава 3 «Процессы и задания» содержит подробное описание реализации процессов в Windows и различных операций с ними. Также здесь описаны задания как механизмы управления наборами процессов и поддержки контейнеров Windows.
- Глава 4 «Потоки» рассказывает об управлении, планировании и других операциях с программными потоками в Windows.
- Глава 5 «Управление памятью» показывает, как диспетчер памяти работает с физической и виртуальной памятью и как процессы и драйверы могут использовать память.

- Глава 6 «Подсистема ввода/вывода» показывает, как работает система ввода/вывода в Windows и как она интегрируется с драйверами устройств для формирования механизмов работы с периферийными устройствами ввода/вывода.
- Глава 7 «Безопасность» посвящена различным механизмам безопасности, встроенным в Windows. В частности, здесь рассматриваются защитные меры, которые сейчас стали частью системы борьбы с эксплойтами.

Благодарности

Прежде всего мы хотим поблагодарить Павла Йосифовича (Pavel Yosifovich), присоединившегося к этому проекту. Его участие сыграло исключительно важную роль для выпуска книги; только благодаря многим ночам, проведенным им за изучением подробностей Windows и сбором информации об изменениях в шести выпусках Windows, эта книга появилась на свет.

Книга не содержала бы столько технических подробностей и не была бы настолько точной, если бы не участие и поддержка ключевых участников группы разработки Microsoft Windows и других экспертов из Microsoft. Мы хотим поблагодарить людей, которые предоставили технические рецензии и/или исходный материал для книги или просто обеспечивали поддержку и помогали авторам: Акила Шриниваасан (Akila Srinivasan), Алессандро Пилотти (Alessandro Pilotti), Андреа Аллиеви (Andrea Allievi), Энди Лурс (Andy Luhrs), Арун Кишан (Arun Kishan), Бен Хиллис (Ben Hillis), Билл Мессмер (Bill Messmer), Крис Клейнханс (Chris Kleynhans), Дипу Томас (Deepu Thomas), Юджин Бак (Eugene Bak), Джейсон Ширк (Jason Shirk), Джеремайя Кокс (Jeremiah Cox), Джо Бялек (Joe Bialek), Джон Ламберт (John Lambert), Джон Ленто (John Lento), Джон Берри (Jon Berry), Кай Су (Kai Hsu), Кен Джонсон (Ken Johnson), Лэнди Ванг (Landy Wang), Логан Габриэль (Logan Gabriel), Люк Ким (Luke Kim), Мэтт Миллер (Matt Miller), Мэтью Вулман (Matthew Woolman), Мехмет Иган (Mehmet Iyigun), Мишель Бержерон (Michelle Bergeron), Минсан Ким (Minsang Kim), Мохамед Мансур (Mohamed Mansour), Нэйт Уорфилд (Nate Warfield), Нирадж Сингх (Neeraj Singh), Ник Джадж (Nick Judge), Павел Лебединский (Pavel Lebedynskiy), Рич Тернер (Rich Turner), Сарухан Карадемир (Saruhan Karademir), Саймон Поуп (Simon Pope), Стивен Финниган (Stephen Finnigan) и Стивен Хафнагел (Stephen Hufnagel).

Хочется еще раз поблагодарить Ильфака Гуилфанова (Ilfak Guilfanov) из компании Hex-Rays (*www.hex-rays.com*) за лицензии IDA Pro Advanced и Hex-Rays, которые были предоставлены Алексу Ионеску (Alex Ionescu) более 10 лет назад, благодаря чему он смог ускорить свой анализ ядра Windows, а также за непрестанную поддержку и разработку средств декомпиляции, сделавших возможным написание этой книги без доступа к исходному коду.

И наконец, авторы хотят поблагодарить замечательный коллектив Microsoft Press, воплотивший эту книгу в реальность. Для Девона Масгрейва (Devon Musgrave)

этот проект стал последним на должности рецензента издательства, а Кейт Шауп (Kate Shoup) стала руководителем проекта. Шон Монингстар (Shawn Morningstar), Келли Тэлбот (Kelly Talbot) и Корина Лебеджоара (Corina Lebegioara) также внесли свой вклад в качество книги.

Список опечаток и качество книги

Мы приложили все усилия к тому, чтобы обеспечить точность материала книги и прилагаемого контента. Все ошибки, о которых было сообщено с момента публикации книги, указаны на сайте Microsoft Press по адресу:

https://aka.ms/winint7ed/errata

Если вы обнаружите ошибку, которая еще не указана в списке, вы можете сообщить нам об этом на той же странице.

Если вам понадобится дополнительная поддержка, обратитесь в службу поддержки Microsoft Press по адресу *mspinput@microsoft.com*.

Пожалуйста, учтите, что поддержка продуктов Microsoft по указанным выше адресам не предоставляется.

От издательства

Обращаем ваше внимание, что в данном издании были оставлены оригинальные скриншоты (с англоязычными названиями элементов интерфейса). В тексте книги использованы русские названия согласно официальной терминологии Microsoft (*https://www.microsoft.com/ru-ru/language*), а в скобках приведены англоязычные термины. Такой подход позволит вам легко ориентироваться в книге и использовать ее с любыми настройками ОС.

Оригинальное седьмое издание книги планируется выпустить в виде двух частей (как и шестое издание), поэтому на страницах данной книги вы найдете ссылки на вторую часть, которая (на момент выпуска первого тиража русскоязычного перевода) еще не была написана.

Ваши замечания, предложения, вопросы отправляйте по адресу comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На веб-сайте издательства www.piter.com вы найдете подробную информацию о наших книгах.

Глава 1
Концепции и средства

В этой главе будут представлены ключевые концепции и термины операционной системы (ОС) Microsoft Windows: Windows API, процессы, программные потоки, виртуальная память, режим ядра и пользовательский режим, объекты, дескрипторы, безопасность и реестр. Также мы рассмотрим некоторые средства, которые могут использоваться для исследования внутреннего строения Windows, например отладчик ядра, системный монитор и важнейшие программы из пакета Windows Sysinternals (*http://www.microsoft.com/technet/sysinternals*). Кроме того, мы объясним, как использовать пакеты Windows Driver Kit (WDK) и Windows Software Development Kit (SDK) для получения дальнейшей информации о внутреннем строении Windows.

После прочтения этой главы проверьте себя: все ли вы поняли? Если вы не усвоите материал этой главы, то не поймете материал оставшейся части книги.

Версии операционной системы Windows

В книге рассматриваются самые последние версии ОС Microsoft Windows для клиента и сервера: Windows 10 (32-разрядная для x86 и ARM, 64-разрядная для x64) и Windows Server 2016 R2 (существует только в 64-разрядной версии). Материал книги относится ко всем версиям, если в тексте явно не указано обратное. Для справки в табл. 1.1 перечислены названия продуктов семейства Windows, их внутренние номера версий и даты релиза.

Начиная с Windows 7, нумерация версий перестала быть очевидной. Системе был присвоен номер версии 6.1 вместо 7. Из-за популярности Windows XP, когда в Windows Vista номер версии был повышен до 6.0, в некоторых приложениях проверка версии ОС стала работать некорректно — разработчики проверяли, что основная версия больше или равна 5, а дополнительная версия больше или равна 1; в Windows Vista это условие не выполнялось. Компания Microsoft усвоила урок и решила оставить основную версию 6 с повышением дополнительной версии до 2 (больше 1), чтобы свести к минимуму подобные несовместимости. Впрочем, в Windows 10 номер версии был обновлен до 10.0.

Таблица 1.1. Релизы операционной системы Windows

Название продукта	Внутренний номер версии	Дата релиза
Windows NT 3.1	3.1	июль 1993 г.
Windows NT 3.5	3.5	сентябрь 1994 г.
Windows NT 3.51	3.51	май 1995 г.
Windows NT 4.0	4.0	июль 1996 г.
Windows 2000	5.0	декабрь 1999 г.
Windows XP	5.1	август 2001 г.
Windows Server 2003	5.2	март 2003 г.
Windows Server 2003 R2	5.2	декабрь 2005 г.
Windows Vista	6.0	январь 2007 г.
Windows Server 2008	6.0 (Service Pack 1)	март 2008 г.
Windows 7	6.1	октябрь 2009 г.
Windows Server 2008 R2	6.1	октябрь 2009 г.
Windows 8	6.2	октябрь 2012 г.
Windows Server 2012	6.2	октябрь 2012 г.
Windows 8.1	6.3	октябрь 2013 г.
Windows Server 2012 R2	6.3	октябрь 2013 г.
Windows 10	10.0 (сборка 10240)	июль 2015 г.
Windows 10 версия 1511	10.0 (сборка 10586)	ноябрь 2015 г.
Windows 10 версия 1607 (Anniversary Update)	10.0 (сборка 14393)	июль 2016 г.
Windows Server 2016	10.0 (сборка 14393)	октябрь 2016 г.

ПРИМЕЧАНИЕ Начиная с Windows 8, функция Windows API GetVersionEx по умолчанию возвращает номер версии 6.2 (Windows 8) независимо от фактической версии ОС. (Эта функция также объявлена устаревшей.) Это сделано для того, чтобы свести к минимуму проблемы совместимости; кроме того, такой подход показывает, что проверка версии ОС в большинстве случаев не является оптимальным решением. Дело в том, что некоторые компоненты могут устанавливаться «раньше времени», без согласования с официальным выпуском Windows. Тем не менее, если вам нужно узнать фактическую версию ОС, вы можете получить ее при помощи функции VerifyVersionInfo или более новых вспомогательных API проверки версий: `IsWindows8OrGreater`, `IsWindows8Point1OrGreater`, `IsWindows10OrGreater`, `IsWindowsServer` и т. д. Кроме того, совместимость с разными операционными системами ОС может быть обозначена в манифесте исполняемого файла, что приводит к изменению результатов вызова этой функции. (За подробностями обращайтесь к разделу «Загрузчик образа» главы 3.)

Для просмотра информации о версии Windows можно воспользоваться программой командной строки ver или ее графической версией winver. Вот как выглядит результат выполнения winver в Windows 10 Enterprise версии 1511:

Графическая версия также выводит номер сборки Windows (10586.218 в данном примере); он может быть полезен для участников программы предварительной оценки Windows Insiders (зарегистрировавшихся для получения ранних ознакомительных версий Windows). Информация может пригодиться и для управления обновлениями безопасности, потому что в ней указан уровень установленного исправления.

Windows 10 и будущие версии Windows

С выходом Windows 10 компания Microsoft объявила, что обновление Windows отныне пойдет в более быстром темпе. Официальной версии «Windows 11» не будет; вместо этого Windows Update (или другая модель корпоративного обслуживания) будет обновлять существующую версию Windows 10 до новой версии. На момент написания книги произошло два обновления: в ноябре 2015 года (известное как «версия 1511» по году и месяцу) и в июле 2015 (версия 1607, известная под маркетинговым названием *«Anniversary Update»*).

ПРИМЕЧАНИЕ На внутреннем уровне Microsoft продолжает строить версии Windows «волнами». Например, исходной версии Windows 10 было присвоено кодовое название Threshold 1, тогда как обновление в ноябре 2015 года называлось Threshold 2. Следующие три фазы обновлений назывались Redstone 1 (версия 1607), Redstone 2 и Redstone 3.

Windows 10 и OneCore

За прошедшие годы появилось несколько разновидностей Windows. Кроме «массовых» версий Windows, работающих на PC, существует ответвление Windows 2000 для игровой приставки Xbox 360. Система Windows Phone 7 использует версию на базе Windows CE (ОС реального времени от Microsoft). Конечно, со-

провождение и расширение всех этих кодовых баз создавало немало сложностей, поэтому в Microsoft решили свести воедино разные ядра и базовые двоичные модули платформенной поддержки. Все началось с использования в Windows 8 и Windows Phone 8 общего ядра (а в Windows 8.1 и Windows Phone 8.1 — единого Windows Runtime API). В Windows 10 слияние завершилось; единая платформа, получившая название OneCore, работает на PC, смартфонах, игровой приставке Xbox One, HoloLens и устройствах IoT (Internet of Things), таких как Raspberry Pi 2.

Понятно, что все эти форм-факторы устройств очень сильно отличаются друг от друга. Некоторые функции на ряде устройств просто отсутствуют. Например, поддержка мыши или физической клавиатуры на устройстве HoloLens не имеет смысла, поэтому вряд ли можно ожидать наличия соответствующих компонентов в версии Windows 10 для таких устройств. Однако ядро, драйверы и двоичные файлы базовой платформы фактически остаются неизменными (с настройками на уровне реестра и/или политики по соображениям производительности или другим причинам). Пример такой политики приведен в разделе «Наборы API-функций» главы 3 «Процессы и задания».

В книге будет рассматриваться внутреннее строение ядра OneCore независимо от того, на каком устройстве оно выполняется. Тем не менее описанные в книге эксперименты предполагают использование настольного компьютера с мышью и клавиатурой, и воспроизвести их на других устройствах (скажем, на телефоне или Xbox One) непросто, а иногда и официально невозможно.

Фундаментальные концепции и термины

В этом разделе представлены фундаментальные концепции Windows, необходимые для понимания материала остальных глав книги. Многие концепции, такие как программные потоки и виртуальная память, подробно рассматриваются в последующих главах.

Windows API

Windows API (Application Programming Interface) — программный интерфейс пользовательского режима для ОС семейства Windows. До появления 64-разрядных версий Windows программный интерфейс 32-разрядных версий ОС Windows назывался *Win32 API* в отличие от исходного 16-разрядного Windows API, программного интерфейса для исходных 16-разрядных версий Windows. В этой книге термин *Windows API* относится как к 32-разрядным, так и 64-разрядным программным интерфейсам Windows.

ПРИМЕЧАНИЕ Иногда мы используем термин Win32 API вместо Windows API. В любом случае он все равно относится как к 32-разрядным, так и к 64-разрядным версиям.

ПРИМЕЧАНИЕ Описание Windows API содержится в документации Windows SDK. (См. раздел «Windows SDK» далее в этой главе.) Документация доступна бесплатно по адресу *https://developer.microsoft.com/en-us/windows/desktop/develop*. Также она включается во все уровни подписки MSDN (Microsoft Developer Network), программы поддержки разработчиков от компании Microsoft. Отличное введение в программирование с использованием базового Windows API представлено в книге «Windows via C/C++, Fifth Edition» Джеффри Рихтера (Jeffrey Richter) и Кристофа Насарра (Christophe Nasarre) (Microsoft Press, 2007).

Разновидности Windows API

Изначально Windows API состоял только из функций в стиле C. Сегодня разработчикам доступны тысячи функций. Выбор языка C был естественным на момент появления Windows, потому что он был своего рода «наименьшим общим кратным» (т. е. написанный на нем код также мог использоваться из других языков) и он был достаточно низкоуровневым для предоставления сервиса ОС. Обратной стороной было огромное количество функций в сочетании с недостаточной последовательностью выбора имен и отсутствием логических группировок (вроде пространств имен C++). Эти сложности привели к тому, что в некоторых новых API используется другой механизм — модель COM (Component Object Model, «модель составного объекта»).

Технология COM изначально создавалась для того, чтобы приложения Microsoft Office могли взаимодействовать друг с другом и передавать данные между документами (например, чтобы в документ Word можно было вставить диаграмму Excel или презентацию PowerPoint). Эта функциональность получила название OLE (Object Linking and Embedding, «связывание и внедрение объектов»). Сначала технология OLE была реализована на базе старого механизма передачи сообщений в Windows, который назывался DDE (Dynamic Data Exchange, «динамический обмен данными»). Технология DDE обладала рядом непреодолимых ограничений, поэтому был разработан новый коммуникационный механизм — COM. Более того, в первом варианте, который был представлен около 1993 года, технология COM изначально называлась OLE 2.

COM базируется на двух основополагающих принципах. Во-первых, клиенты взаимодействуют с объектами (которые иногда называются серверными объектами COM) через интерфейсы — четко определенные контракты с набором логически связанных методов, сгруппированных посредством механизма диспетчеризации по виртуальным таблицам (этот же механизм обычно применяется компиляторами C++ для реализации диспетчеризации виртуальных функций). Таким образом обеспечивается двоичная совместимость и снимаются проблемы с декорированием имен компилятором. Соответственно, такие методы могут вызываться из многих других языков (и компиляторов), включая C, C++, Visual Basic, языки .NET, Delphi и т. д. Второй принцип — динамическая загрузка компонентов (вместо статической компоновки с клиентом).

Термин «*сервер COM*» обычно относится к DLL-библиотеке или исполняемому файлу (EXE), в котором реализованы классы COM. COM также содержит ряд

важных функций, связанных с безопасностью, межпроцессным маршалингом, потоковой моделью и т. д. Подробное знакомство с COM выходит за рамки книги; отличное описание можно найти в книге Дона Бокса (Don Box) «Essential COM» (Addison-Wesley, 1998).

ПРИМЕЧАНИЕ Среди примеров API, доступ к которым осуществляется через COM, можно назвать DirectShow, Windows Media Foundation, DirectX, DirectComposition, WIC (Windows Imaging Component) и BITS (Background Intelligent Transfer Service).

Windows Runtime

В Windows 8 появились новый API и исполнительная среда поддержки *Windows Runtime* (иногда используется сокращение WinRT — не путайте с Windows RT, версии ОС Windows на базе ARM, поддержка которой была прекращена). Windows Runtime состоит из платформенных сервисов, предназначенных для разработчиков приложений *Windows Apps* (ранее также использовались термины *Metro Apps*, *Modern Apps*, *Immersive Apps* и *Windows Store Apps*). Приложения Windows Apps подходят для разных форм-факторов устройств, от миниатюрных IoT-устройств до телефонов, планшетов, ноутбуков, настольных систем, и даже таких устройств, как Xbox One и Microsoft HoloLens.

С точки зрения API платформа WinRT строится на базе COM, добавляя в базовую инфраструктуру COM различные расширения. Например, в WinRT доступны полные метаданные типов (хранящиеся в файле WINMD и основанные на формате метаданных .NET), расширяющие аналогичную концепцию библиотек типов в COM. С точки зрения архитектуры API она обладает намного большей целостностью, чем классические функции Windows API: в ней реализованы иерархии пространств имен, последовательная схема назначения имен и паттерны программирования.

Приложения Windows Apps строятся по новым правилам и не похожи на привычные приложения Windows (которые теперь называются настольными приложениями Windows или классическими приложениями Windows). Эти правила описаны в главе 9 «Механизмы управления» части 2[1].

Отношения между различными API и приложениями не столь прямолинейны. Настольные приложения могут использовать подмножество WinRT API. И наоборот, приложения Windows Apps могут использовать подмножество Win32 и COM API. За подробной информацией о том, какие API доступны для каждой платформы приложений, обращайтесь к документации MSDN.

Однако обратите внимание на то, что на базовом двоичном уровне WinRT API все равно строится на основе унаследованных двоичных файлов и API Windows, даже если доступность некоторых API не документирована и официально не под-

[1] Здесь и далее используются ссылки на вторую часть книги, которая на момент выпуска первого тиража русскоязычного перевода еще не опубликована (находится в работе у авторов).

держивается. Это не новый «машинный» API для системы, а ситуация напоминает то, как .NET строится на основе традиционного Windows API.

Приложения, написанные на C++, C# (и других языках .NET) и JavaScript, могут легко использовать WinRT API благодаря языковым проекциям, разработанным для этих платформ. Для C++ компания Microsoft создала нестандартное расширение C++/CX, которое упрощает работу с типами WinRT. Обычная прослойка взаимодействия COM для .NET (с некоторыми расширениями исполнительной среды) позволяет любому языку .NET использовать WinRT API естественно и просто, как если бы это была чистая среда .NET. Чтобы разработчики JavaScript могли работать с WinRT, было разработано расширение *WinJS*, хотя для построения пользовательских интерфейсов разработчикам JavaScript все равно приходится использовать HTML.

ПРИМЕЧАНИЕ Хотя разметка HTML может использоваться в приложениях Windows Apps, они все равно остаются локальными клиентскими приложениями, а не веб-приложениями, загружаемыми с веб-сервера.

.NET Framework

.NET Framework является частью Windows. В табл. 1.2 перечислены версии .NET Framework, устанавливаемые в составе разных версий Windows. Впрочем, новые версии .NET Framework могут устанавливаться и в старых версиях ОС.

Таблица 1.2. Версии .NET Framework, устанавливаемые по умолчанию в Windows

Версия Windows	Версия .NET Framework
Windows 8	4.5
Windows 8.1	4.5.1
Windows 10	4.6
Windows 10 версия 1511	4.6.1
Windows 10 версия 1607	4.6.2

.NET Framework состоит из двух основных компонентов:

- **CLR (Common Language Runtime).** Исполнительная среда .NET; включает JIT-компилятор (Just-In-Time) для преобразования инструкций языка CIL (Common Intermediate Language) в низкоуровневый язык машинных команд процессора, уборщик мусора, систему проверки типов, безопасность обращения к коду и т. д. Среда реализована в виде внутрипроцессного сервера COM (DLL) и использует различные средства, предоставляемые Windows API.
- **.NET Framework Class Library (FCL).** Обширная подборка типов, реализующих функциональность, часто используемую в клиентских и серверных при-

ложениях, – средства пользовательского интерфейса, поддержка сети, работа с базами данных и т. д.

Среда .NET Framework, предоставляющая эти и другие возможности, включая высокоуровневые языки программирования (C#, Visual Basic, F#) и вспомогательные средства, повышает производительность труда разработчика, а также безопасность и надежность приложений. На рис. 1.1 изображены отношения между .NET Framework и ОС Windows.

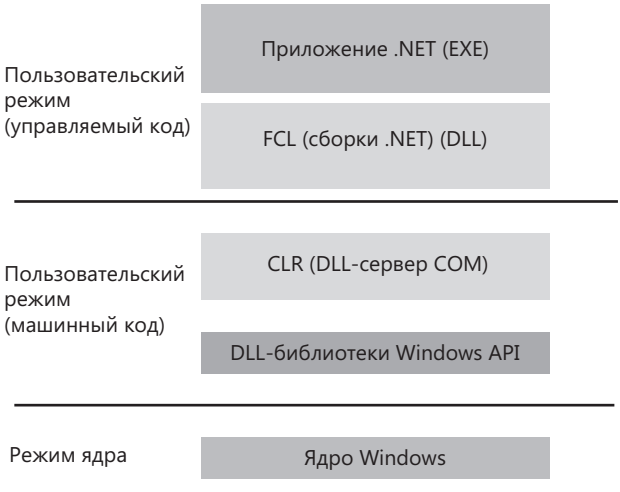


Рис. 1.1. Отношения между .NET и ОС Windows

Службы, функции и процедуры

Некоторые термины в документации пользователя и разработчика Windows имеют разный смысл в разных контекстах. Например, словом «*служба*» (service) может обозначаться код ОС, который может вызываться извне, драйвер устройства или серверный процесс. В следующем списке указано, какой смысл имеют некоторые термины из книги.

- **Функции Windows API.** Документированные, открытые для вызова процедуры Windows API. Примеры — `CreateProcess`, `CreateFile` и `GetMessage`.
- **Системные вызовы (низкоуровневые системные функции).** Недокументированные сервисные функции ОС, которые могут вызываться из пользовательского режима. Например, `NtCreateUserProcess` — внутренняя системная функция, вызываемая функцией Windows `CteateProcess` для создания нового процесса.
- **Вспомогательные функции ядра (процедуры).** Подпрограммы ОС Windows, которые могут вызываться только из режима ядра (см. далее в этой главе). Например, процедура `ExAllocatePoolWithTag` вызывается драйверами устройств для выделения памяти из системного пула Windows (так называемой *кучи*).

- **Службы Windows.** Процессы, запускаемые диспетчером служб Windows. Например, служба планировщика задач выполняется в процессе пользовательского режима, поддерживающего команду schtasks (аналог команд UNIX at и cron). (Заметим, что хотя в системном реестре драйверы устройств Windows определяются как «службы», в книге термин в этом контексте использоваться не будет.)
- **Библиотеки динамической компоновки (DLL).** Подпрограммы, предназначенные для внешнего вызова и объединенные в двоичные файлы, которые могут динамически загружаться приложениями, использующими эти подпрограммы. Примеры: Msvcrt.dll (библиотека времени выполнения C) и Kernel32.dll (одна из библиотек подсистемы Windows API). Приложения и компоненты пользовательского режима Windows широко используют DLL-библиотеки. Их преимущество перед статическими библиотеками заключается в том, что DLL могут совместно использоваться приложениями; система Windows позаботится о том, чтобы в памяти находилась только одна копия кода DLL-библиотеки для всех приложений, работающих с ней. Обратите внимание: библиотечные сборки .NET компилируются в формат DLL, но без неуправляемых экспортируемых подпрограмм. Вместо этого CLR разбирает откомпилированные метаданные для обращения к соответствующим типам и членам.

Процессы

На первый взгляд может показаться, что процессы и программы похожи, но в действительности между ними существуют принципиальные различия. *Программа* — статическая последовательность команд, тогда как *процесс* — контейнер для набора ресурсов, используемых для выполнения программы. На верхнем уровне абстракции процесс Windows включает следующие компоненты:

- **закрытое виртуальное адресное пространство** — множество адресов виртуальной памяти, которая может использоваться процессом;
- **исполняемая программа**, которая определяет первоначальный код и данные и отображается в виртуальное адресное пространство процесса;
- **список открытых дескрипторов** для различных системных ресурсов (семафоров, объектов синхронизации портов, файлов и т. д.), доступных для всех программных потоков в процессе;
- **контекст безопасности** — *маркер доступа* (access token), который идентифицирует пользователя, группы безопасности, привилегии, состояние виртуализации UAC (User Account Control), сеанс и ограниченное состояние учетной записи пользователя, связанное с процессом, а также идентификатор контейнера приложения и связанная с ним информация изоляции;
- **идентификатор процесса** — уникальный идентификатор, который является частью *идентификатора клиента*;

- **по меньшей мере один программный поток** (thread). Пустые процессы теоретически могут существовать, но особой пользы не принесут.

Существуют различные программы для просмотра (и изменения) процессов и информации процессов. Описанные ниже эксперименты показывают, какую информацию о процессах можно получить при помощи некоторых средств такого рода. Многие из этих средств включены в систему Windows, средства отладки для Windows и Windows SDK; другие являются самостоятельными программами из пакета Sysinternals. Многие программы выводят перекрывающиеся подмножества базовых данных процессов и программных потоков, которые иногда обозначаются разными именами.

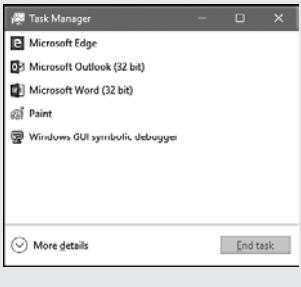
Вероятно, для просмотра информации о процессах чаще всего используется диспетчер задач (Task Manager). (Выбор названия программы выглядит немного странно, так как в ядре Windows не существует понятия «задачи» (task).) Следующий эксперимент демонстрирует некоторые базовые возможности диспетчера задач.

ЭКСПЕРИМЕНТ: ПРОСМОТР ИНФОРМАЦИИ ПРОЦЕССОВ В ДИСПЕТЧЕРЕ ЗАДАЧ

Диспетчер задач, встроенный в Windows, выдает простой список процессов в системе. Диспетчер задач можно запустить четырьмя способами:

- Нажмите Ctrl+Shift+Esc.
- Щелкните правой кнопкой мыши на панели задач и выберите команду диспетчер задач (Start Task Manager).
- Нажмите Ctrl+Alt+Del и щелкните на кнопке Запустить диспетчер задач (Start Task Manager).
- Запустите исполняемый файл Taskmgr.exe.

При первом запуске диспетчер задач работает в «кратком» режиме, в котором выводятся только процессы, имеющие видимое окно верхнего уровня, как на следующем снимке экрана:

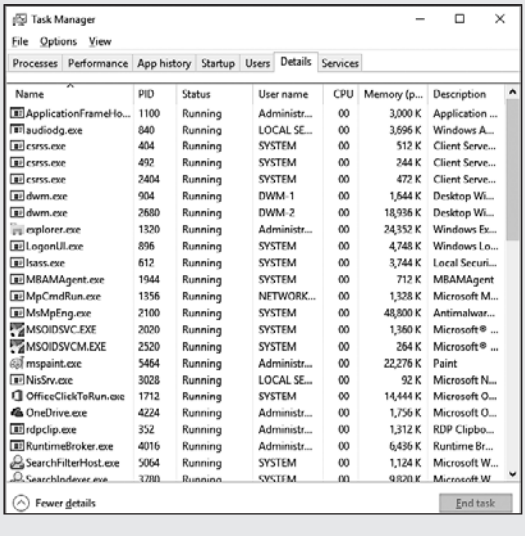


В этом режиме ваши возможности сильно ограничены, поэтому щелкните на кнопке Подробнее (More Details), чтобы открыть полное представление диспетчера задач. По умолчанию выбирается вкладка Процессы (Processes):

На вкладке Процессы (Processes) выводится список процессов, состоящий из четырех столбцов: ЦП (CPU), Память (Memory), Диск (Disk) и Сеть (Network). Чтобы добавить в список другие столбцы, щелкните правой кнопкой мыши на заголовке. Также доступны столбцы Process (Image) name, ИД процесса (Process ID), Тип (Type), Состояние (Status), Издатель (Publisher) и Командная строка (Command Line). Некоторые процессы можно дополнительно развернуть с выводом информации о видимых окнах верхнего уровня, созданных процессом.

Чтобы получить еще больше информации о процессе, щелкните на кнопке Подробнее (Details). Также можно щелкнуть правой кнопкой мыши на процессе и выбрать команду Подробнее (Go to Details), чтобы переключиться на вкладку Подробности (Details) и выбрать этот конкретный процесс.

ПРИМЕЧАНИЕ Вкладка Процессы (Processes) диспетчера задач Windows 7 приблизительно эквивалентна вкладке Подробности (Details) диспетчера задач Windows 8+. На вкладке Приложения (Applications) диспетчера задач Windows 7 выводятся видимые окна верхнего уровня, а не процессы как таковые. В новом диспетчере задач Windows 8+ эта информация теперь содержится на вкладке Процессы (Processes).



На вкладке Подробности (Details) также выводятся процессы, но в более компактном виде. На ней нет информации об окнах, созданных процессами, но больше столбцов с разнообразными данными.

Обратите внимание: процессы идентифицируются по имени образа, экземплярами которого они являются. В отличие от некоторых объектов Windows, процессам не могут присваиваться глобальные имена. Для вывода дополнительной информации щелкните правой кнопкой мыши на заголовке и щелкните на кнопке Выбрать столбцы (Select Columns). Открывается список столбцов, который выглядит так:

Некоторые важнейшие столбцы:

- **Потоки** (Threads) — в этом столбце выводится количество программных потоков в каждом процессе. Это число обычно не меньше 1, так как невозможно напрямую создать процесс, не содержащий ни одного потока (к тому же такой процесс будет бесполезен). Если в списке присутствует процесс с 0 потоков, обычно это означает, что процесс не удается удалить по какой-либо причине — чаще всего из-за ошибки в коде драйвера.
- **Дескрипторы** (Handles) — в этом столбце выводится количество дескрипторов объектов ядра, открытых программными потоками, выполняемыми в процессе. (Дескрипторы рассматриваются далее в этой главе, а также более подробно в главе 8 части 2.)
- **Состояние** (Status) — с этим столбцом дело обстоит сложнее. Для процессов, не имеющих пользовательского интерфейса, в нем обычно выводится значение Выполняется (Running), хотя все потоки могут чего-то ожидать, например сигнала о состоянии объекта ядра или завершения операции ввода/вывода. Другое возможное значение — Приостановлен (Suspended) — встречается в том случае, если все потоки процесса находятся в приостановленном состоянии. Вряд ли это произойдет в результате деятельности самого процесса, хотя может быть следствием вызова для процесса недокументированной функции API `NtSuspendProcess`, чаще всего при помощи служебной программы (например, программы Process Explorer, описанной ниже). Для процессов, создающих пользовательский интерфейс, состояние Выполняется (Running) означает, что пользовательский интерфейс реагирует на действия пользователя. Иначе говоря, поток, создавший окно (или окна), ожидает пользовательского ввода (с технической точки зрения — очереди сообщений, связанной с потоком). Состояние приостановки возможно и без пользовательского интерфейса, но

для приложений Windows Apps (на платформе Windows Runtime) приостановка обычно происходит тогда, когда приложение уходит с первого плана из-за того, что оно было свернуто пользователем. Такие процессы приостанавливаются через 5 секунд, чтобы они не поглощали ресурсы процессора или сети, а новое приложение первого плана могло получить в свое распоряжение все ресурсы машины. Это особенно важно для устройств с питанием от аккумулятора, таких как планшеты или телефоны. Эти и другие сопутствующие механизмы более подробно описаны в главе 9 части 2. Третье возможное значение в столбце Состояние (Status) — Не отвечает (Not Responding). Оно возникает в том случае, если программный поток процесса, создавший пользовательский интерфейс, не проверял свою очередь сообщений на предмет UI-событий по крайней мере 5 секунд. Процесс (а на самом деле поток, которому принадлежит окно) может быть занят работой, интенсивно загружающей процессор, или может ожидать чего-то совершенно иного (например, завершения операции ввода/вывода). В любом случае пользовательский интерфейс «зависает», а Windows сообщает об этом, затеняя такое окно (или окна) и выводя в столбце Состояние (Status) значение Не отвечает (Not Responding).

Каждый процесс также содержит идентификатор своего родительского процесса или процесса-создателя (они могут совпадать, но это не обязательно). Если родитель не существует, то информация не обновляется. Следовательно, процесс может хранить идентификатор несуществующего родителя; это не создает проблем, потому что система не полагается на актуальность этой информации. В программе Process Explorer учитывается время запуска родительского процесса, чтобы избежать связывания дочернего процесса на основании уже переназначенного идентификатора процесса. Следующий эксперимент демонстрирует это поведение.

ПРИМЕЧАНИЕ Почему процесс-родитель может не совпадать с процессом-создателем? В некоторых случаях процессы, которые на первый взгляд были созданы определенным пользовательским приложением, могут использовать участие вспомогательного процесса (процесса-брокера), отвечающего за вызовы API создания процессов. В таких случаях указание процесса-брокера в качестве создателя будет создавать путаницу (и даже ошибки при использовании наследования адресного пространства или дескрипторов), поэтому требуется «смена родителя». Один из примеров такого рода приведен в главе 7 «Безопасность».

ЭКСПЕРИМЕНТ: ПРОСМОТР ДЕРЕВА ПРОЦЕССОВ

Большинство служебных программ не выводит идентификаторы родителей или создателей процессов. Для получения этой информации можно воспользоваться Системным монитором (или запросить идентификатор процесса-создателя из программного кода). Также можно воспользоваться программой Tlist.exe из средств отладки Windows и запросить вывод дерева процессов при помощи ключа `/t`. Пример вывода команды `tlist /t`:

```
System Process (0)
System (4)
  smss.exe (360)
csrss.exe (460)
```