

АНДРЕЙ ПОПОВ

Введение в Windows PowerShell

**Работа в новой оболочке
командной строки
Windows PowerShell**

**Описание языка
PowerShell**

**Использование объектов
.NET, WMI, ADSI и COM**

**Интеграция
с командными файлами
cmd.exe и сценариями
WSH**

Андрей Попов

Введение в Windows PowerShell

Санкт-Петербург

«БХВ-Петербург»

2009

УДК 681.3.06
ББК 32.973.26-018.2
П58

Попов А. В.

П58 Введение в Windows PowerShell. — СПб.: БХВ-Петербург, 2009. — 464 с.: ил. — (Системный администратор)

ISBN 978-5-9775-0283-2

Рассматривается новая объектно-ориентированная оболочка командной строки Microsoft Windows PowerShell и ее возможности для автоматизации повседневных задач администрирования. Описываются основные элементы и конструкции языка PowerShell. Приводятся примеры использования объектов .NET, WMI, ADSI и COM. Обсуждаются вопросы совместного использования PowerShell, командных файлов интерпретатора cmd.exe и сценариев Windows Script Host. Даются примеры решения с помощью PowerShell задач администратора Windows.

Для администраторов, программистов и опытных пользователей Windows

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Юрий Якубович</i>
Компьютерная верстка	<i>Натальи Караваевой</i>
Корректор	<i>Виктория Пиотровская</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.08.08.

Формат 70×100¹/₁₆. Печать офсетная. Усл. печ. л. 37,41.

Тираж 2000 экз. Заказ №

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

Отпечатано с готовых диапозитивов

в ГУП "Типография "Наука"

199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0283-2

© Попов А. В., 2008

© Оформление, издательство "БХВ-Петербург", 2008

Оглавление

- Введение 1**
 - Что это за книга и для кого она предназначена 2
 - Структура книги 3
 - Принятые в книге соглашения 5
- Часть I. Изучаем PowerShell 7**
 - Глава 1. Windows PowerShell — результат развития технологий автоматизации 9**
 - Предшественники PowerShell в мире Windows 9
 - Оболочка командной строки command.com/cmd.exe 12
 - Сервер сценариев Windows Script Host (WSH)..... 14
 - Оболочка WMI Command-line (WMIC) 17
 - Причины и цели создания оболочки PowerShell..... 18
 - Отличие PowerShell от других оболочек — ориентация на объекты 22
 - Глава 2. Первые шаги в PowerShell. Основные понятия..... 25**
 - Загрузка и установка PowerShell 25
 - Запуск оболочки 26
 - Работают ли знакомые команды? 26
 - Вычисление выражений 28
 - Типы команд PowerShell 29
 - Командлеты 30
 - Функции..... 35
 - Сценарии 36
 - Внешние исполняемые файлы 36
 - Псевдонимы команд..... 36
 - Диски PowerShell..... 40
 - Провайдеры PowerShell..... 41

Навигация по дискам PowerShell	43
Просмотр содержимого дисков и каталогов	44
Создание дисков	46
Глава 3. Приемы работы в оболочке	48
Редактирование в командном окне PowerShell	48
Автоматическое завершение команд	50
Справочная система PowerShell	52
Получение справки о командлетах	53
Справочная информация, не связанная с командлетами	55
История команд в сеансе работы	58
Протоколирование действий в сеансе работы	62
Глава 4. Настройка оболочки	65
Настройка ярлыка PowerShell	65
Программное изменение свойств консоли PowerShell	67
Цвета текста и фона	68
Заголовок командного окна	68
Размеры командного окна	69
Приглашение командной строки	70
Настройка пользовательских профилей	72
Политики выполнения сценариев	75
Глава 5. Работа с объектами	78
Конвейеризация объектов в PowerShell	78
Просмотр структуры объектов (командлет <i>Get-Member</i>)	80
Фильтрация объектов (командлет <i>Where-Object</i>)	82
Сортировка объектов (командлет <i>Sort-Object</i>)	85
Выделение объектов и свойств (командлет <i>Select-Object</i>)	87
Выполнение произвольных действий над объектами в конвейере (командлет <i>ForEach-Object</i>)	90
Группировка объектов (командлет <i>Group-Object</i>)	91
Измерение характеристик объектов (командлет <i>Measure-Object</i>)	92
Вызов статических методов	93
Управление выводом команд в PowerShell	95
Форматирование выводимой информации	96
Перенаправление выводимой информации	99
Глава 6. Переменные, массивы и хэш-таблицы	102
Числовые и символьные литералы	102
Числовые литералы	102
Символьные строки	103

Переменные PowerShell	107
Переменные оболочки PowerShell	108
Пользовательские переменные. Типы переменных	111
Переменные среды Windows	115
Массивы в PowerShell	117
Обращение к элементам массива	117
Операции с массивом	118
Хэш-таблицы (ассоциативные массивы)	121
Операции с хэш-таблицей	124
Глава 7. Операторы и управляющие инструкции	126
Арифметические операторы	126
Оператор сложения	127
Оператор умножения	129
Операторы вычитания, деления и остатка от деления	130
Операторы присваивания	131
Операторы сравнения	133
Операторы проверки на соответствие шаблону	135
Логические операторы	138
Управляющие инструкции языка PowerShell	139
Инструкция <i>If ... ElseIf ... Else</i>	139
Цикл <i>While</i>	140
Цикл <i>Do ... While</i>	141
Цикл <i>For</i>	141
Цикл <i>ForEach</i>	142
Метки циклов, инструкции <i>Break</i> и <i>Continue</i>	145
Инструкция <i>Switch</i>	146
Глава 8. Функции, фильтры и сценарии	152
Функции в PowerShell	152
Обработка аргументов функций с помощью переменной <i>\$Args</i>	153
Формальные параметры функций	155
Возвращаемые значения	160
Функции внутри конвейера команд	162
Фильтры в PowerShell	163
Функции в качестве командлетов	164
Сценарии PowerShell	166
Создание и запуск сценариев PowerShell	167
Передача аргументов в сценарии	169
Выход из сценариев	170
Оформление сценариев. Комментарии	171

Глава 9. Обработка ошибок и отладка	175
Обработка ошибок	175
Объект ErrorRecord и поток ошибок	176
Сохранение объектов, соответствующих ошибкам	179
Мониторинг возникновения ошибок	182
Режимы обработок ошибок	183
Обработка "критических" ошибок (исключений)	185
Отладка сценариев	187
Вывод диагностических сообщений	187
Командлет Set-PSDebug	188
Трассировка выполнения команд	189
Пошаговое выполнение команд	191
Вложенная командная строка и точки прерывания	191
 Часть II. Используем PowerShell.....	195
 Глава 10. Доступ из PowerShell к внешним объектам (COM, WMI, .NET и ADSI)	197
Работа с COM-объектами	197
Внешние серверы автоматизации на примере Microsoft Office	202
Доступ к объектам WMI	208
Подключение к подсистеме WMI. Получение списка классов	209
Получение объектов WMI	211
Выполнение WQL-запросов	214
Использование объектов .NET	215
Доступ к службе каталогов ADSI	218
 Глава 11. Работа с файловой системой	222
Навигация в файловой системе	222
Получение списка файлов и каталогов	223
Определение размера каталогов	227
Создание файлов и каталогов	228
Чтение и просмотр содержимого файлов	229
Запись файлов	230
Копирование файлов и каталогов	232
Переименование и перемещение файлов и каталогов	235
Удаление файлов и каталогов	236
Поиск текста в файлах	237
Замена текста в файлах	241

Глава 12. Управление процессами и службами	243
Управление процессами	243
Просмотр списка процессов	245
Определение библиотек, используемых процессом	250
Остановка процессов	252
Запуск процессов, изменение приоритетов выполнения	253
Завершение неотвечающих процессов	255
Управление службами	255
Просмотр списка служб	256
Остановка и приостановка служб	258
Запуск и перезапуск служб	259
Изменение параметров службы	259
Глава 13. Работа с системным реестром.....	261
Структура реестра	261
Просмотр локального реестра	263
Просмотр удаленного реестра	265
Модификация реестра	268
Создание нового раздела	269
Копирование разделов	269
Переименование раздела	270
Удаление раздела	270
Создание параметра	270
Изменение значения параметра	271
Переименование параметра	272
Копирование параметров	272
Очистка значения параметра	273
Удаление параметра	273
Глава 14. Работа с журналами событий	275
Инструменты для обработки журналов событий	277
Список журналов событий на локальном компьютере	279
Список журналов событий на удаленном компьютере	281
Просмотр событий из локального журнала	282
Вывод событий определенного типа	285
Отбор событий по идентификатору	286
Отбор событий по датам	287
Группировка событий по источнику возникновения	288
Просмотр событий из удаленного журнала	289

Настройка журналов событий	292
Установка максимального размера журналов	292
Установка режима хранения журналов	292
Очистка журнала	293
Глава 15. Управление рабочими станциями. Получение и анализ системной информации.....	295
Завершение сеанса пользователя	295
Перезагрузка и выключение компьютера	297
Получение информации о BIOS.....	298
Вывод списка команд, выполняемых при загрузке системы	299
Вывод свойств операционной системы.....	301
Вывод списка установленных программных продуктов.....	303
Вывод списка установленных обновлений операционной системы	306
Глава 16. Инвентаризация оборудования	309
Получение информации о физической памяти	309
Преобразование отчета в формат HTML.....	311
Получение информации о процессорах	313
Получение списка устройств Plug-and-Play	316
Получение информации о звуковой карте	319
Получение информации о видеокарте	320
Получение информации о сетевых адаптерах	323
Глава 17. Настройка сетевых параметров. Работа с электронной почтой	325
Получение и настройка сетевых параметров	325
Получение списка IP-адресов компьютера	326
Вывод параметров протокола TCP/IP	327
Настройка DHCP.....	332
Отправка сообщений по электронной почте	338
Глава 18. PowerShell, cmd.exe и VBScript: совместное использование	341
Сравнение языков PowerShell и cmd.exe.....	342
Различия в синтаксисе команд	342
Работа с переменными	344
Использование циклов	346
Вывод текста и запуск программ	347
Запуск команд cmd.exe из PowerShell	347

Сравнение языков PowerShell и VBScript.....	349
Обращение к функциям, командам и методам	349
Работа с переменными, массивами и объектами.....	351
Использование символьных строк.....	351
Прочие замечания по синтаксису.....	352
Аналоги PowerShell для функций VBScript.....	353
Математические функции.....	353
Символьные функции.....	355
Функции для работы с датами и временем	361
Использование из PowerShell кода VBScript.....	368
Использование из PowerShell кода JScript.....	370
Заключение.....	371

П Р И Л О Ж Е Н И Я373

Приложение 1. Объектная модель WMI.....375

Общая структура WMI.....	376
Ядро WMI	377
Провайдеры WMI	378
Менеджер объектов CIM	379
Репозиторий CIM. Пространства имен.....	381
Путь к классам и объектам CIM.....	384
Безопасность при работе с WMI.....	385
Структура классов WMI	389
Основные типы классов CIM.....	389
Свойства классов WMI.....	391
Методы классов WMI.....	397
Квалификаторы классов, свойств и методов	399
Интерактивная работа с объектами WMI	403
Тестер WMI (WBEMTest).....	403
Административные утилиты WMI (WMI Tools)	404

Приложение 2. Полезные COM-объекты и примеры их использования.... 411

Управление проводником Windows с помощью объекта <i>Shell.Application</i>	411
Отображение специальных окон Проводника.....	413
Вызов элементов панели управления	418
Управление открытыми окнами.....	420

Использование объектов Windows Script Host	423
Работа с ресурсами локальной сети (объект <i>WScript.Network</i>)	423
Вывод информационного окна (объект <i>WScript.Shell</i>)	428
Переключение между приложениями, имитация нажатий клавиш (объект <i>WScript.Shell</i>)	430
Доступ к специальным папкам Windows (объект <i>WScript.Shell</i>)	436
Удаление некорректных ярлыков (объект <i>WScript.Shell</i>)	438
Ссылки на ресурсы Интернета	441
Сайты компании Microsoft	441
Другие сайты	441
Группы новостей	442
Блоги.....	442
Список литературы	443
Предметный указатель	445

Введение

Windows PowerShell — это новая оболочка командной строки и среда выполнения сценариев для операционной системы Windows, разработанная компанией Microsoft относительно недавно (финальный релиз версии 1.0 этой оболочки был выпущен в ноябре 2006 года). Главной задачей, которую ставили перед собой разработчики, было создание среды составления сценариев, которая наилучшим образом подходила бы для современных версий операционной системы Windows и была бы более функциональной, расширяемой и простой в использовании, чем какой-либо аналогичный продукт для любой другой операционной системы. В первую очередь эта среда должна была подходить для решения задач, стоящих перед системными администраторами, а также удовлетворять требованиям разработчиков программного обеспечения, предоставляя им средства для быстрой реализации интерфейсов управления создаваемыми приложениями. Сейчас уже можно сказать, что продукт оказался очень продуманным и удачным, а его мощные возможности сочетаются с простотой использования. Актуальность изучения и использования PowerShell вызвана тем, что компания Microsoft в настоящее время позиционирует эту оболочку как основной инструмент управления операционной системой и рядом разработанных ею приложений (PowerShell официально включен в качестве стандартного компонента в операционную систему Windows Server 2008, а также используется в таких продуктах Microsoft, как Exchange Server 2007, System Center Operations Manager 2007, System Center Data Protection Manager V2 и System Center Virtual Machine Manager).

За рубежом выпущено уже несколько книг, посвященных различным аспектам PowerShell (отдельно выделим здесь электронные книги [8] и [9], которые можно свободно загрузить из Интернета). Также много ресурсов имеется в Интернете, отметим некоторые из них (адреса этих ресурсов приведены в конце книги в *разд. "Ссылки на ресурсы Интернета")*:

- на сайте Microsoft поддерживается активно обновляемый раздел TechNet Script Center, содержащий примеры сценариев PowerShell, ссылки на статьи и вебкасты, посвященные этому продукту;

- разработчики PowerShell ведут в Интернете блоги, в которых они делятся информацией о возможностях PowerShell;
- действует группа новостей, посвященная PowerShell.

К сожалению, с систематизированной информацией о PowerShell на русском языке дело обстоит несколько хуже, особенно не хватает подробного описания нового языка программирования PowerShell. Что можно сейчас найти в Интернете по этой теме? На сайте компании Microsoft доступны локализованная версия PowerShell (встроенная справка переведена на русский язык) и пакет документации на русском языке (введение в новую оболочку командной строки и краткий обзор языка PowerShell). Периодически переводы статей по PowerShell появляются в журналах "Windows & .Net Magazine/RE" (<http://www.osp.ru>) и "TechNet Magazine" (<http://www.microsoft.com/technet/technetmag>). Отдельно хочется сказать спасибо Д. Сотникову, А. Бешкову и В. Гусеву, ведущим блоги и проводящим вебкасты по тематике PowerShell.

Что это за книга и для кого она предназначена

При написании этой книги была поставлена цель восполнить недостаток информации о PowerShell на русском языке. В частности, хотелось решить следующие задачи:

- пояснить лежащие в основе PowerShell базовые механизмы работы с объектами и описать основные конструкции и элементы языка PowerShell (за основу взята великолепно написанная монография [1], которую можно порекомендовать всем, кто захочет более глубоко изучить PowerShell);
- помочь пользователям, знакомым с командным интерпретатором cmd.exe и сервером сценариев WSH, начать работу с новым средством автоматизации Windows;
- привести практические примеры использования PowerShell для решения типичных задач администратора операционной системы Windows.

Сама постановка этих задач подразумевает некоторое знакомство читателя с компьютером. Поэтому книга ориентирована на администраторов информационных систем на базе Windows и обычных опытных пользователей, желающих изучить новую мощную оболочку командной строки от Microsoft и автоматизировать с помощью нее свою повседневную работу на компьютере.

Структура книги

Книга состоит из двух частей. В первой части, озаглавленной *"Изучаем PowerShell"*, последовательно рассматриваются основные концепции новой оболочки командной строки, элементы и конструкции языка PowerShell.

В *первой главе* обсуждаются возможности и недостатки средств автоматизации работы в операционной системе Windows, существовавших до появления PowerShell (оболочки cmd.exe и WMIC, сервера сценариев Windows Script Host), приводятся причины и цели разработки новой оболочки командной строки. Особо выделяется основное отличие PowerShell от других распространенных оболочек командной строки — ориентация на работу с объектами, а не с потоком текста.

Во *второй главе* приводятся инструкции по установке и запуску PowerShell, описываются типы команд, используемые в данной оболочке. Обсуждаются понятия псевдонимов команд и дисков PowerShell.

В *третьей главе* изучаются приемы интерактивной работы в оболочке PowerShell и способы обращения к встроенной справочной системе.

В *четвертой главе* рассматриваются вопросы настройки интерфейса оболочки, пользовательских профилей и политик выполнения сценариев PowerShell.

В *пятой главе* речь идет об основном механизме PowerShell — конвейеризации объектов, и в качестве примеров описываются базовые манипуляции с объектами, которые можно выполнять в этой оболочке (фильтрация, сортировка, группировка и т. д.). Разбираются механизмы управления выводом команд PowerShell (форматирование и перенаправление результирующей информации).

Шестая глава посвящена изучению основных структур данных, использующихся в PowerShell (константы, переменные, массивы и хэш-таблицы).

В *седьмой главе* рассматриваются основные операторы и управляющие конструкции языка PowerShell.

В *восьмой главе* обсуждаются вопросы написания программных модулей (функций и сценариев) на встроенном в оболочку языке.

Девятая глава посвящена имеющимся в PowerShell средствам обработки ошибок и отладки сценариев.

Во второй части книги, *"Используем PowerShell"*, показаны примеры применения интерактивных команд и сценариев PowerShell для решения практических задач.

В *десятой главе* рассматриваются вопросы доступа из PowerShell к различным объектным моделям и технологиям, которые поддерживаются операци-

онной системой Windows (COM-объекты, подсистема Windows Management Instrumentations, служба каталогов ADSI, объекты платформы .NET). Приводятся примеры управления приложениями пакета Microsoft Office.

В *одиннадцатой главе* на примерах показывается, как с помощью команд PowerShell выполнять основные операции с объектами файловой системы.

В *двенадцатой главе* изучаются команды, позволяющие управлять процессами и службами.

В *тринадцатой главе* рассматриваются приемы работы из оболочки PowerShell с системным реестром Windows.

В *четырнадцатой главе* приводятся примеры работы с журналами событий Windows, использующие команды PowerShell.

В *пятнадцатой главе* обсуждаются задачи получения системной информации о настройках операционных систем и установленного программного обеспечения на локальном и удаленных компьютерах, а также команды для управления рабочими станциями в сети.

В *шестнадцатой главе* предложены команды PowerShell, которые можно использовать для составления HTML-отчетов об аппаратных компонентах компьютеров в сети.

В *семнадцатой главе* рассматриваются вопросы получения и настройки параметров сетевого протокола TCP/IP. Приводятся примеры сценариев на языке PowerShell, которые отправляют сообщения по электронной почте.

В *восемнадцатой главе* обсуждаются проблемы совместного использования различных средств автоматизации Windows (команд интерпретатора cmd.exe, сценариев WSH на языках VBScript или JScript, команд и сценариев PowerShell). Проводится сравнение языков cmd.exe и VBScript с языком PowerShell, приводятся примеры использования из PowerShell кода VBScript и JScript.

Кроме основного материала, в книге имеется два приложения. В *приложении 1* рассматривается объектная модель Windows Management Instrumentation (WMI), которая широко используется для выполнения различных задач администратора операционной системы. Предоставление доступа к этой модели из командной строки было одной из основных целей при разработке PowerShell.

В *приложении 2* описываются некоторые COM-объекты, которые могут оказаться полезными для администраторов и пользователей Windows, и даются примеры использования этих объектов из PowerShell.

Принятые в книге соглашения

Оболочка PowerShell — это интерактивная среда, поэтому во многих примерах показаны как команды, вводимые пользователем, так и ответ на них, генерируемый системой. Перед командой указывается приглашение PowerShell, обычно выглядящее как `PS C:\>`. Сама вводимая команда выделяется полужирным шрифтом, например `Get-Process`. В следующих нескольких строках приводится текст, возвращаемый системой в ответ на введенную команду, например:

PS C:\> **Get-Process**

Handles	NPM(K)	PM(K)	WS(K)	VM(M)	CPU(s)	Id	ProcessName
-----	-----	-----	-----	-----	-----	--	-----
99	5	1116	692	32	0.07	232	alg
39	1	364	500	17	0.14	1636	ati2evxx
57	3	1028	1408	30	0.38	376	atiptaxx
412	6	2128	3600	26	6.50	808	csrss
64	3	812	1484	29	0.19	316	ctfmon
386	13	13748	14448	77	16.11	1848	explorer
171	5	4512	584	44	0.20	428	GoogleT...
0	0	0	16	0		0	Idle
151	4	2908	992	41	1.05	412	kav
. . .							

Многоточие здесь указывает на то, что для экономии места приведены не все строки, возвращаемые командой `Get-Process`.

Иногда вводимые команды могут разбиваться на несколько строк. В этих случаях перед каждой дополнительной строкой команды указываются символы `>>`, например:

```
PS C:\> dir *.tmp | ForEach-Object {  
>> $arr=$_.Name.split(".");  
>> $newname=$arr[0]+".new";  
>> ren $_.FullName $newname -PassThru;  
>> }  
>>
```


Каталог: Microsoft.PowerShell.Core\FileSystem::C:\

Mode	LastWriteTime	Length	Name	
----	-----	-----	----	
-a---	17.06.2008	11:39	0	3.new
. . .				

При описании операторов, функций и методов объектов используются стандартные соглашения. Названия подставляемых параметров и аргументов набраны курсивом, а необязательные параметры заключены в квадратные скобки [], например:

```
CreateObject(strProgID [,strPrefix])
```



Ч А С Т Ь I

Изучаем PowerShell



Глава 1

Windows PowerShell — результат развития технологий автоматизации

Прежде чем приступить непосредственно к изучению оболочки командной строки Windows PowerShell, попробуем ответить на ряд вопросов. Для чего, собственно, компании Microsoft потребовалось создавать этот совершенно новый инструмент и язык программирования? Какую пользу он может принести обычным пользователям и системным администраторам? Почему недостаточно было существующих средств?

Чтобы понять это, посмотрим, как в операционной системе Windows обстояло дело с *автоматизацией работы*, то есть решением различных задач в автоматическом режиме, без участия человека, до появления PowerShell.

Предшественники PowerShell в мире Windows

В настоящее время графический интерфейс Windows стал настолько привычным, что многие пользователи и начинающие администраторы даже не задумываются об альтернативных способах управления операционной системой с помощью *командной строки* (command line) и различных *сценариев* (scripts). Зачастую они просто не знают о тех преимуществах, которые дают эти инструменты с точки зрения автоматизации работы.

Подобная ситуация обусловлена тем, что исторически командная строка всегда была слабым местом операционной системы Windows (по сравнению с UNIX-системами). Причиной этого, прежде всего, является то, что компания Microsoft изначально ориентировалась на широкую аудиторию неискушенных пользователей, не желающих особо вникать в технические детали выполнения тех или иных действий в системе. Поэтому основные усилия разработчиков операционной системы направлялись на улучшение графической оболочки для более комфортной работы непрофессионалов, а не на создание рабочей среды для специалистов или опытных пользователей.

Как показало время, с коммерческой точки зрения на рынке персональных (домашних или офисных) компьютеров эта стратегия оказалась более чем успешной: миллионы людей используют графический интерфейс Windows для запуска нужных им программ, работы в офисных пакетах, просмотра фильмов и т. п. Да и управлять одним Windows-сервером сегодня несложно: операционная система предлагает удобные графические средства для настройки различных параметров и выполнения ежедневных администраторских задач, а с помощью службы терминалов легко можно работать на удаленном сервере, физически расположенном хоть на другом континенте.

Однако подобная модель управления не является масштабируемой: если с помощью стандартных графических инструментов администрировать не один, а десять серверов, то последовательность изменений настроек в диалоговых окнах придется повторить десять раз. Следовательно, в этом случае остро встает вопрос об автоматизации выполнения рутинных операций (например, проведения инвентаризации оборудования и программного обеспечения, мониторинга работы служб, анализа журналов событий и т. д.) на множестве компьютеров. Помочь в этом могут либо специальные (как правило, тяжеловесные и недешевые) приложения типа Microsoft Systems Management Server (SMS), либо сценарии, которые пишутся администраторами самостоятельно (на языке оболочки командной строки или на специальных языках сценариев) и поддерживаются непосредственно операционной системой, без установки сторонних программных продуктов.

Поэтому для профессионала, занимающегося администрированием информационных систем на базе Windows, знание возможностей командной строки, сценариев и технологий автоматизации, поддерживаемых данной операционной системой, просто необходимо.

При этом, однако, неправильно было бы думать, что командная строка или сценарии нужны только администраторам. Ведь рутинные ежедневные задачи пользователей (связанные, например, с копированием или архивированием файлов, подключением или отключением сетевых ресурсов и т. п.), которые обычно выполняются с помощью графического интерфейса проводника Windows, можно полностью самостоятельно автоматизировать, написав нехитрый командный файл, состоящий всего из нескольких строчек! Однако для человека, не знающего основные команды Windows и такие базовые возможности операционной системы, как перенаправление ввода/вывода и конвейеризация команд, некоторые простейшие задачи могут показаться нетривиальными. Попробуйте, например, пользуясь только графическими средствами, сформировать файл, содержащий имена файлов из всех подкаталогов какого-либо каталога! А ведь для этого достаточно выполнить единст-

венную команду `dir` (с определенными ключами) и перенаправить вывод этой команды в нужный текстовый файл. Например, следующая команда создаст текстовый файл `c:\list_mp3.txt`, в котором будут записаны имена всех файлов с расширением `mp3`, находящихся в каталоге `c:\music` или в каком-либо его подкаталоге:

```
dir /s /b c:\music\*.mp3 > c:\list_mp3.txt
```

Задумаясь теперь, каким же нам хотелось бы видеть инструмент для автоматизации работы в операционной системе, какими возможностями он должен обладать? Желательно, чтобы в нем было реализовано следующее:

- работа в разных версиях операционной системы (в идеальном случае во всех) без установки какого-либо дополнительного программного обеспечения;
- интеграция с командной строкой (непосредственное выполнение вводимых с клавиатуры команд);
- согласованный и непротиворечивый синтаксис команд и утилит;
- наличие подробной встроенной справки по командам с примерами использования;
- возможность выполнения сценариев, составленных на простом для изучения языке;
- возможность использования всех технологий, поддерживаемых операционной системой.

В UNIX-системах в качестве инструмента автоматизации выступает стандартная оболочка (`sh`) или ее модификации (`bash`, `ksh`, `csk` и т. д.), причем этот аспект операционной системы стандартизирован в рамках POSIX (стандарт мобильных систем).

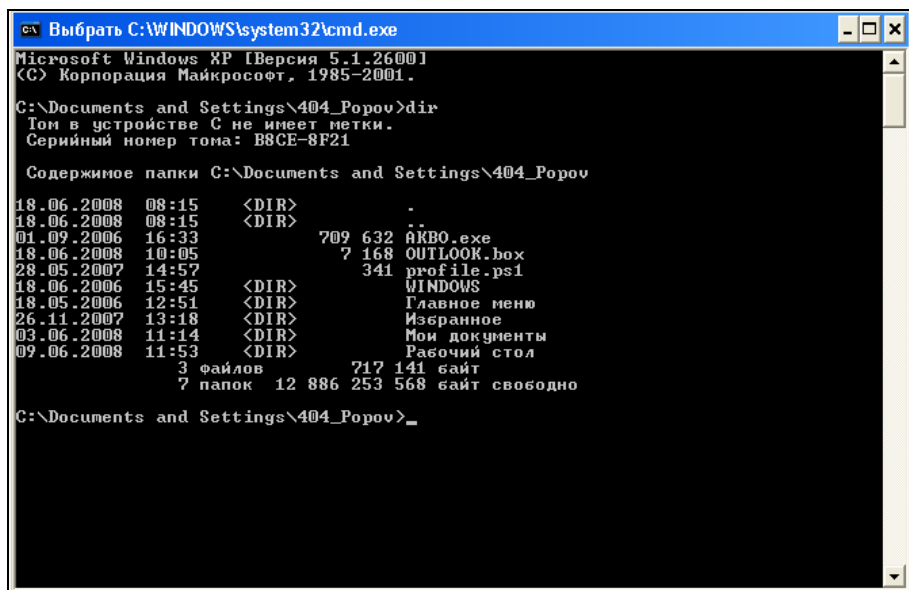
В операционной системе Windows дело обстоит сложнее. На сегодняшний день одного "идеального" средства автоматизации, удовлетворяющего сразу всем перечисленным выше требованиям, в Windows нет. В последних версиях операционной системы одновременно поддерживаются несколько стандартных инструментов автоматизации, сильно отличающихся друг от друга: оболочка командной строки `cmd.exe`, среда выполнения сценариев Windows Script Host (WSH), оболочка WMI Command-line (WMIC) и, наконец, новинка и предмет нашего изучения — оболочка Microsoft PowerShell. Поэтому администратору или пользователю Windows приходится выбирать, каким именно подходом воспользоваться для решения определенной задачи, а для этого желательно иметь четкое представление о сильных и слабых сторонах всех доступных средств автоматизации.

Оболочка командной строки `command.com/cmd.exe`

Во всех версиях операционной системы Windows поддерживается интерактивная *оболочка командной строки* (command shell), и по умолчанию устанавливается определенный набор утилит командной строки (количество и состав этих утилит зависит от версии операционной системы). Вообще, любую операционную систему можно представить в виде совокупности *ядра системы*, которое имеет доступ к аппаратуре и управляет файлами и процессами, и *оболочки (командного интерпретатора)* с утилитами, которые позволяют пользователю получить доступ к функциональности ядра операционной системы. Механизм работы оболочек в разных системах одинаков: в ответ на приглашение ("подсказку", prompt), выдаваемое находящейся в ожидании оболочкой, пользователь вводит некоторую команду (функциональность этой команды может быть реализована либо самой оболочкой, либо определенной внешней утилитой), оболочка выполняет ее, при необходимости выводя на экран какую-либо информацию, после чего снова выводит приглашение и ожидает ввода следующей команды (рис. 1.1).

ЗАМЕЧАНИЕ

С технической точки зрения оболочка представляет собой построчный интерпретатор простого языка сентенциального (директивного) программирования, в качестве операторов которого могут использоваться исполняемые программы.



```
С:\> Выбрать C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Версия 5.1.2600]
(C) Корпорация Майкрософт, 1985-2001.

C:\Documents and Settings\404_Popov>dir
Том в устройстве C не имеет метки.
Серийный номер тома: B8CE-8F21

Содержимое папки C:\Documents and Settings\404_Popov

18.06.2008  08:15    <DIR>          .
18.06.2008  08:15    <DIR>          ..
01.09.2006  16:33             709 632 AKBO.exe
18.06.2008  10:05             7 168 OUTLOOK.box
28.05.2007  14:57             341 profile.ps1
18.06.2006  15:45    <DIR>          WINDOWS
18.05.2006  12:51    <DIR>          Главное меню
26.11.2007  13:18    <DIR>          Избранное
03.06.2008  11:14    <DIR>          Мои документы
09.06.2008  11:53    <DIR>          Рабочий стол
                3 файлов             717 141 байт
                7 папок      12 886 253 568 байт свободно

C:\Documents and Settings\404_Popov>_
```

Рис. 1.1. Результат выполнения команды `dir` в оболочке `cmd.exe`

Наряду с интерактивным режимом работы оболочки, как правило, поддерживают и пакетный режим, в котором система последовательно выполняет команды, записанные в текстовом файле-сценарии. Оболочка Windows не является исключением, с точки зрения программирования язык командных файлов Windows может быть охарактеризован следующим образом:

- реализация сентенциальной (директивной) парадигмы программирования;
- выполнение в режиме построчной интерпретации;
- наличие управляющих конструкций;
- поддержка нескольких видов циклов (в том числе специальных циклов для обработки текстовых файлов);
- наличие оператора присваивания (установки значения переменной);
- возможность использования внешних программ (команд) операционной системы в качестве операторов и обработки их кодов возврата;
- наличие нетипизированных переменных, которые декларируются первым упоминанием (значение переменных могут интерпретироваться как числа и использоваться в выражениях целочисленной арифметики).

Начиная с версии Windows NT, оболочка командной строки представляется интерпретатором `cmd.exe`, который расширяет возможности оболочки `com-mand.com` операционной системы MS-DOS. В свою очередь функциональность командного интерпретатора `command.com` была позаимствована из операционной системы CP/M, командный интерпретатор которой представлял собой значительно упрощенный и урезанный вариант оболочки UNIX-систем.

Таким образом, оболочка командной строки MS-DOS изначально уступала UNIX-оболочкам по удобству работы и развитости языка сценариев. В командной оболочке Windows NT (`cmd.exe`), несмотря на все сделанные улучшения, не удалось преодолеть данное отставание ни в режиме интерактивной работы (например, в `cmd.exe` отсутствует поддержка псевдонимов для длинных названий команд и не реализовано автоматическое завершение команд при вводе их с клавиатуры), ни в синтаксисе или возможностях языка командных файлов.

Ситуация усугублялась тем, что Windows всегда проигрывала UNIX-системам в количестве и функциональных возможностях стандартных (не требующих дополнительной установки) утилит командной строки, а также в качестве и полноте встроенной справочной системы по командам оболочки.

ЗАМЕЧАНИЕ

Для того чтобы прочитать встроенную справку для определенной команды, нужно вызвать эту команду с ключом `/?` (например, `хсору /?`). Общий справочник по командной строке находится в файле `%WinDir%\Help\Ntcmds.chm`.

На практике проблему отсутствия нужной функциональности у стандартных команд приходится решать либо с помощью утилит пакета Windows Resource Kit для соответствующей версии операционной системы, либо путем поиска подходящей утилиты сторонних производителей. Кроме того, в Windows можно пользоваться POSIX-совместимыми утилитами и оболочками с помощью пакета Microsoft Services For UNIX (SFU). Данный продукт разрабатывался еще для Windows NT и первоначально не входил в состав операционной системы, его нужно было приобретать за отдельную плату. В дальнейшем пакет SFU стал бесплатным и даже был включен в состав операционной системы Windows Server 2003 R2.

Итак, учитывая все сказанное ранее, мы можем сделать следующий вывод: оболочка командной строки `cmd.exe` и командные файлы — наиболее универсальные и простые в изучении средства автоматизации работы в Windows, доступные во всех версиях операционной системы. При этом, однако, оболочка `cmd.exe` и командные файлы существенно проигрывают аналогичным инструментам в UNIX-системах и не обеспечивают доступ к объектным моделям, поддерживаемым операционной системой (COM, WMI, .NET).

Сервер сценариев Windows Script Host (WSH)

Следующим шагом в развитии средств и технологий автоматизации в операционной системе Windows стало появление сервера сценариев Windows Script Host (WSH). Этот инструмент разработан для всех версий Windows и позволяет непосредственно в операционной системе выполнять сценарии на полноценных языках сценариев (по умолчанию, VBScript и JScript), которые до этого были доступны только внутри HTML-страниц и работали в контексте безопасности веб-браузера (в силу этого подобные сценарии, например, могли не иметь доступа к файловой системе локального компьютера).

По сравнению с командными файлами интерпретатора `cmd.exe` сценарии WSH имеют несколько преимуществ.

Во-первых, VBScript и JScript — это полноценные алгоритмические языки, имеющие встроенные функции и методы для обработки символьных строк, выполнения математических операций, обработки исключительных ситуаций и т. д.; кроме того, для написания сценариев WSH может использоваться любой другой язык сценариев (например, широко распространенный в UNIX-системах Perl), для которого установлен соответствующий модуль поддержки.

Во-вторых, WSH поддерживает несколько собственных объектов, свойства и методы которых позволяют решать некоторые часто возникающие повседневные задачи администратора операционной системы: работа с сетевыми

ресурсами, переменными среды, системным реестром, ярлыками и специальными папками Windows, запуск и управление работой других приложений. Например, в сценарии MakeShortcut.vbs с помощью объекта WshShell создается ярлык на сетевой ресурс: сайт компании Microsoft (листинг 1.1).

Листинг 1.1. Создание ярлыка из сценария (VBScript)

```

'*****
' * Имя: MakeShortcut.vbs
' * Язык: VBScript
' * Описание: Создание ярлыков из сценария
'*****
Dim WshShell,oUrlLink
' Создаем объект WshShell
Set WshShell=WScript.CreateObject("WScript.Shell")
' Создаем ярлык на сетевой ресурс
Set oUrlLink = WshShell.CreateShortcut("Microsoft Web Site.URL")
' Устанавливаем URL
oUrlLink.TargetPath = "http://www.microsoft.com"
' Сохраняем ярлык
oUrlLink.Save
'***** Конец *****

```

В-третьих, из сценариев WSH можно обращаться к службам любых приложений-серверов автоматизации, которые регистрируют в операционной системе свои объекты (скажем, программ из пакета Microsoft Office). Например, в сценарии PrintInWord.vbs происходит подключение к серверу автоматизации Microsoft Word и вывод строк текста в окно этого приложения (листинг 1.2).

Листинг 1.2. Использование сервера автоматизации Microsoft Word (VBScript)

```

'*****
' * Имя: PrintInWord.vbs
' * Язык: VBScript
' * Описание: Использование из сценария внешнего объекта
'               автоматизации (Microsoft Word)
'*****
Option Explicit

Dim WA,WD,Sel ' Объявляем переменные

```

```
'Создаем объект-приложение Microsoft Word
Set WA=WScript.CreateObject("Word.Application")
' Можно было использовать конструкцию
' Set WA=CreateObject("Word.Application")

Set WD=WA.Documents.Add 'Создаем новый документ (объект Document)
WA.Visible=true ' Делаем Word видимым
Set Sel=WA.Selection 'Создаем объект Selection
Sel.Font.Size=14 'Устанавливаем размер шрифта
Sel.ParagraphFormat.Alignment=1 'Выравнивание по центру
Sel.Font.Bold=true 'Устанавливаем полужирный шрифт
Sel.TypeText "Привет!" & vbCrLf 'Печатаем строку текста
Sel.Font.Bold=false 'Отменяем полужирный шрифт
Sel.ParagraphFormat.Alignment=0 'Выравнивание по левому краю
'Печатаем строку текста
Sel.TypeText "Эти строки напечатаны с помощью WSH."
WD.PrintOut 'Выводим документ на принтер
'***** Конец *****
```

Наконец, сценарии WSH позволяют работать с объектами информационной модели Windows Management Instrumentation (WMI), обеспечивающей программный интерфейс управления всеми компонентами операционной модели, а также с объектами службы каталогов Active Directory Service Interface (ADSI) (объектная модель WMI подробно обсуждается в *приложении 1*).

Следует также отметить, что технология WSH поддерживается в Windows уже довольно давно, в Интернете (в том числе на сайте Microsoft) можно найти множество готовых сценариев, выполняющих ту или иную операцию, и при определенных навыках и знаниях быстро "подогнать" эти сценарии под свои конкретные задачи.

Поговорим теперь о слабых местах WSH. Прежде всего, сам по себе WSH — это только среда выполнения сценариев, а не оболочка; WSH не интегрирован с командной строкой, то есть отсутствует режим, в котором можно было вводить команды с клавиатуры и сразу видеть результат их выполнения.

Большим минусом для WSH является то, что в операционной системе по умолчанию нет полноценной подробной справочной информации по объектам WSH и языкам VBScript/JScript (документацию приходится искать в Интернете на сайте Microsoft). Другими словами, если вы, например, не помните синтаксис определенной команды VBScript/JScript или точное название свойства объекта WSH, под рукой у вас нет распечатанной документации,

а компьютер не имеет выхода в Интернет, то написать корректный сценарий вам просто не удастся. (В данном аспекте командные файлы более универсальны, так как практически у всех команд есть, по крайней мере, встроенное описание используемых ими ключей, а в операционной системе имеется справочный файл с информацией обо всех стандартных командах.)

Наконец, сценарии WSH представляют собой довольно серьезную потенциальную угрозу с точки зрения безопасности, известно большое количество вирусов, использующих WSH для выполнения деструктивных действий.

Таким образом, можно дать следующую общую оценку: сценарии WSH — это универсальный инструмент, который в любой версии операционной системы Windows позволяет решать задачи автоматизации практически любой степени сложности, но требует при этом большой работы по изучению самих языков сценариев и ряда смежных технологий управления операционной системой (WMI, ADSI и т. п.).

Оболочка WMI Command-line (WMIC)

Как уже упоминалось, в операционной системе Windows поддерживается информационная модель Windows Management Instrumentation (WMI), которая занимает важное место среди технологий, инструментов и средств автоматизации. В основе данной технологии лежит схема CIM (Common Information Model), которая представляет физическую и логическую структуры компьютерной системы в виде единой расширяемой объектно-ориентированной информационной модели и определяет единые интерфейсы для получения информации о любом компоненте этой модели.

Изначально работать с WMI можно было либо с помощью специальных графических утилит, либо путем составления довольно сложных сценариев WSH. В состав операционных систем Windows XP и Windows Server 2003 была включена утилита WMIC (WMI Command-line), позволяющая обращаться к подсистеме WMI непосредственно из командной строки. Оболочка WMIC поддерживает навигацию по информационной схеме WMI локального или удаленного компьютера, позволяя выполнять WQL-запросы к классам и объектам WMI. При этом вместо сложных названий классов WMI используются простые псевдонимы, причем можно создавать собственные псевдонимы, что делает информационную схему WMIC расширяемой. Например, классу `Win32_OperatingSystem` соответствует псевдоним `os`. Если набрать в командной строке WMIC команду `os` и нажать <Enter>, то мы увидим на экране свойства операционной системы, установленной на компьютере (рис. 1.2).

По умолчанию WMIC поддерживает около 80 псевдонимов, с помощью которых можно выполнить полторы сотни методов и получить значения множества свойств. Важной особенностью WMIC является то, что вывод команд

может быть организован в различные форматы: на экран, в текстовый файл, в XML- и HTML-документы, в MOF-файл, в текстовый файл с разделителями или в любой другой формат, определяемый пользователем с помощью таблиц стилей XSL (eXtensible Stylesheet Language).

```

C:\WINDOWS\System32\Wbem\wmic.exe
winlogon.exe      winlogon.exe
services.exe      C:\WINDOWS\system32\services.exe
lsass.exe         C:\WINDOWS\system32\lsass.exe
smss.exe          C:\WINDOWS\system32\smss.exe
csrss.exe         C:\WINDOWS\system32\csrss.exe -k netsvcs
explorer.exe      C:\WINDOWS\system32\explorer.exe -k NetworkService
spoolsv.exe       C:\WINDOWS\system32\spoolsv.exe
inetinfo.exe      C:\WINDOWS\system32\inetinfo.exe
wmiprvse.exe      C:\WINDOWS\system32\wmiprvse.exe
explorer.exe      C:\WINDOWS\Explorer.EXE
TrayIcon.exe      "C:\WINDOWS\System32\TrayIcon.exe"
MWProEng.exe      "C:\Program Files\MouseWarePro\MWProEng.exe"
Dict.exe          "C:\Program Files\Bridge to English\Oxford Dictionary\Dict.exe" /
ctfmon.exe        "C:\WINDOWS\System32\ctfmon.exe"
msmsgs.exe        "C:\Program Files\Messenger\msmsgs.exe" /background
sqlmangr.exe      "C:\Program Files\Microsoft SQL Server\80\Tools\Binn\sqlmangr.exe"
WINWORD.EXE       "C:\Program Files\Microsoft Office\Office\Winword.exe" D:\BHU\3.d
wmic.exe          "C:\WINDOWS\System32\Wbem\wmic.exe"
mspaint.exe       "C:\WINDOWS\system32\mspaint.exe"
smss.exe          C:\WINDOWS\System32\smss.exe -k imgsvc

wmic:root\cli>OS
BootDevice      BuildNumber      BuildType      Caption
Device\HarddiskVolume3  2600            Uniprocessor Free  Microsoft Windows XP Professional

wmic:root\cli>

```

Рис. 1.2. Результат выполнения команды OS в оболочке WMIC

Одна команда WMIC может быть применена сразу к нескольким удаленным компьютерам с любой 32-разрядной версией Windows, при этом наличие WMIC на удаленной машине не требуется, необходима только установка ядра WMI и соответствующая настройка прав доступа к WMI. Кроме этого, команды WMI могут использоваться в пакетных файлах Windows, что позволяет простыми средствами автоматизировать работу с WMI на локальных или удаленных компьютерах.

В качестве недостатка WMIC можно отметить отсутствие встроенной полноценной поддержки и обработки событий WMI. Как показало время, оболочка WMIC оказалась не особенно удачной, так как в этом продукте акцент был сделан на функциональные особенности WMI, а не на удобстве работы пользователя.

Причины и цели создания оболочки PowerShell

Итак, к началу XXI века в операционной системе Windows поддерживались три разных инструмента для автоматизации работы: оболочки командной

строки cmd.exe и WMIC, а также сервер сценариев WSH. Зачем же компании Microsoft понадобилась разработка еще одной совершенно новой оболочки командной строки со своим языком сценариев?

Дело в том, что у каждого из перечисленных инструментов автоматизации имелись довольно серьезные недостатки, не позволявшие сказать, что Windows обладает по-настоящему мощным и эффективным средством для работы с командной строкой и написания сценариев (см. табл. 1.1). С одной стороны, функциональности и гибкости языка оболочки cmd.exe было явно недостаточно, а с другой стороны, сценарии WSH, работающие с объектными моделями ADSI и WMI, оказались слишком сложными для пользователей среднего уровня и начинающих администраторов.

Таблица 1.1. Требования к инструменту автоматизации

Требование	cmd.exe	WSH	WMIC
Работа во всех версиях операционной системы без установки дополнительного программного обеспечения	Да	Да	Нет (только Windows XP и выше)
Интеграция с командной строкой	Да	Нет	Да
Согласованный и непротиворечивый синтаксис команд и утилит	Нет	Нет	Да
Поддержка псевдонимов (кратких синонимов) для длинных названий команд	Нет	Нет	Да
Автоматическое завершение команд и имен файлов при вводе их с клавиатуры	Частично (автоматическое завершение имен файлов и папок)	Нет	Нет
Поддержка истории введенных команд с возможностью их повторного вызова, просмотра и редактирования	Да	Нет	Да
Наличие подробной встроенной справки по командам с примерами использования	Частично	Нет	Да