**ЛЕКЦИЯ. POWER SHELL**

До появления PowerShell существовали (*и существуют*) следующие инструменты для автоматизации и администрирования сервисов: [командная строка Windows](https://info-comp.ru/vseowindowsst/53-comandstroka.html) и Windows Script Host. Но у этих технологий есть недостатки.

У командной строки Windows есть возможность выполнять какие-то административные задачи и возможность записать алгоритм действий, сохранив его в виде скрипта ([bat-файла](https://info-comp.ru/obucheniest/81--bat-.html)), при этом можно использовать элементы программирования, например, переменные, условные конструкции и циклы.

Большинство программных продуктов имеет консольный интерфейс, т.е. мы можем управлять программой, используя командную строку, при этом экономя ресурсы за счет отсутствия затрат на работу графического интерфейса. Компания Microsoft для серверной операционной системы Windows Server выпускает редакции без графического интерфейса (*Server Core, в [Windows Server 2016](https://info-comp.ru/sisadminst/543-install-and-review-of-windows-server-2016.html" \o "Установка Windows Server 2016 и обзор новых возможностей) даже есть Nano Server*), но всего этого недостаточно, так как возможности командной строки ограничены, т.е. написать какую-то сложную логику для автоматизации чего-либо мы не сможем, а если и сможем, то на это нам потребуется время и знания.

Также недостатком командной строки является то, что для полноценного управления сервером используются различные консольные утилиты программных продуктов, и поэтому использовать командную строку для администрирования на самом деле сложно, с точки зрения того, что у всех этих утилит разный синтаксис, параметры, методы использования.

Технология Windows Script Host позволяет выполнять все административные задачи, что и командная строка, включая их автоматизацию путем написания [WSH скриптов](https://info-comp.ru/programmirovanie/255-jscript-wsh.html), но здесь уже можем использовать полноценные языки программирования (**VBScript и JScript**), т.е. можно реализовывать сложную логику и алгоритмы. К тому же с помощью WSH мы управляем программными продуктами через объектный интерфейс, другими словами Windows Script Host намного «*круче*» чем командная строка. Но данная технология также не стала тем идеальным инструментом администрирования и автоматизации этого администрирования для системных администраторов, так как Windows Script Host требовал знаний вышеперечисленных языков программирования, что для системных администраторов на самом деле лишнее. Администраторам нужно всего лишь простой инструмент администрирования с возможностью запрограммировать какие-то действия, а углубляться в объектные модели программных продуктов на языках программирования VBScript и JScript им не хочется.

В итоге компании Microsoft необходимо было разработать такой инструмент администрирования для системных администраторов, который бы на 100 процентов удовлетворял все потребности сисадминов как в плане возможностей администрирования и автоматизации, так и в плане удобства и простоты использования, таким образом, появился Windows PowerShell.

Загрузка и установка PowerShell

Оболочка PowerShell может работать не во всех версиях операционной системы Windows, ее можно использовать в Windows XP SP2, Windows Server 2003 SP1 и более поздних версиях. Кроме этого, в системе должна быть установлена платформа .NET Framework 2.0 (в Windows XP данную среду придется устанавливать дополнительно, дистрибутив можно загрузить с сайта Microsoft по адресу http://msdn.microsoft.com/netframework/downloads/updates/default.aspx).

Загрузить PowerShell можно с сайта Microsoft, открыв страницу http://microsoft.com/powershell, где имеются ссылки на файл установки последней версии и пакеты документации на разных языках. Запустив загруженный файл, следуйте указаниям мастера установки.

В 32-разрядных версиях Windows оболочка PowerShell устанавливается по умолчанию в каталог %SystemRoot%\System32\WindowsPowerShell\v1.0.

В 64-разрядных версиях Windows 32-разрядная версия PowerShell устанавливается в каталог %SystemRoot%\SystemWow64\WindowsPowerShell\v1.0,а 64-разрядная версия Windows PowerShell устанавливается в каталог %SystemRoot%\System32\WindowsPowerShell\v1.0.

Выполним первую команду в PowerShell. Пусть это будет что-то знакомое по работе с оболочкой cmd.exe, например, команда dir. Вспомним сначала, как выглядит вывод этой команды в cmd.exe:

C:\Documents and Settings\User> dir

Том в устройстве C имеет метку C

Глава 2. Первые шаги в PowerShell. Основные понятия 27

Серийный номер тома: 44D2-29CE

Содержимое папки C:\Documents and Settings\User

22.01.2008 22:10 <DIR> .

22.01.2008 22:10 <DIR> ..

12.05.2005 16:16 <DIR> DoctorWeb

24.05.2006 20:22 8 304 gsview32.ini

06.11.2004 13:03 <DIR> Phone Browser

04.10.2004 14:33 <DIR> Главное меню

05.12.2007 00:49 <DIR> Избранное

31.10.2007 21:03 <DIR> Мои документы

23.01.2008 11:13 <DIR> Рабочий стол

1 файлов 8 304 байт

8 папок 1 734 508 544 байт свободно

Итак, на экран выводится список файлов и подкаталогов в текущем каталоге, а также некоторая дополнительная информация. Теперь запустим dir в PowerShell (отметим, что в новой оболочке команды по-прежнему обрабатываются без учета регистра):

PS C:\Documents and Settings\User> dir

Каталог: Microsoft.PowerShell.Core\FileSystem::C:\Documents and Settings\

User

Mode LastWriteTime Length Name

---- ------------- ------ ----

d---s 14.12.2007 10:10 Cookies

d---- 12.05.2005 17:16 DoctorWeb

d---- 06.11.2004 13:03 Phone Browser

d---s 22.09.2004 23:49 UserData

d-r-- 04.10.2004 15:33 Главное меню

d-r-- 05.12.2007 0:49 Избранное

d-r-- 31.10.2007 21:03 Мои документы

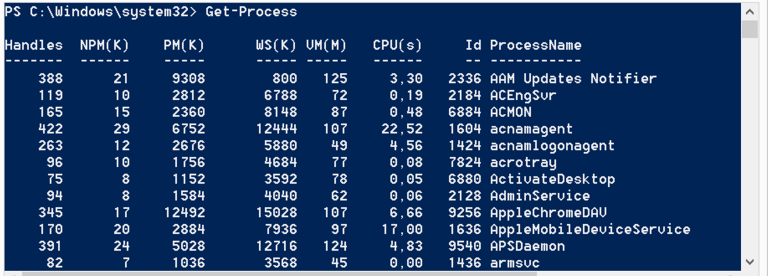
d---- 05.12.2007 10:54 Рабочий стол

-a--- 24.05.2006 21:22 8304 gsview32.ini

Команды dir в оболочках cmd.exe и PowerShell различаются.

Для получения текущих процессов выполним команду:

Get-Process

результат:  
  


Get-Service #для получения статуса служб, запущенных на компьютерах

Get-Content C:\Windows\System32\drivers\etc\hosts #для получения содержимого файла. В данном случае файла hosts

Информацию о всех доступных командах (называют также командлетами) можно получить, введя следующую команду:

Get-Help -Category cmdlet

Для запуска процесса:  
  
Start-Process notepad

Что аналогично записи: 

start notepad

Остановим процесс:

Stop-Process -Name notepad

Следующая команда возвращает процессы, имена которых начинаются с "ex.".

PS> Get-Process -Name ex\*

Handles NPM(K) PM(K) WS(K) VM(M) CPU(s) Id ProcessName

------- ------ ----- ----- ----- ------ -- -----------

234 7 5572 12484 134 2.98 1684 EXCEL

555 15 34500 12384 134 105.25 728 explorer

Поскольку класс System.Diagnostics.Process .NET является основой для процессов Windows PowerShell, он удовлетворяет некоторым соглашениям, используемым System.Diagnostics.Process. Одно из таких соглашений требует, чтобы имя процесса для исполняемого файла никогда не содержало ".exe" в конце имени этого файла.

**Get-Process** также принимает несколько значений для параметра Name.

PS> Get-Process -Name exp\*,power\*

Handles NPM(K) PM(K) WS(K) VM(M) CPU(s) Id ProcessName

------- ------ ----- ----- ----- ------ -- -----------

540 15 35172 48148 141 88.44 408 explorer

605 9 30668 29800 155 7.11 3052 powershell

Параметр ComputerName командлета Get-Process можно использовать для получения процессов на удаленных компьютерах. Например, следующая команда получает процессы PowerShell на локальном (представленным "localhost") и двух удаленных компьютерах.

PS> Get-Process -Name PowerShell -ComputerName localhost, Server01, Server02

Handles NPM(K) PM(K) WS(K) VM(M) CPU(s) Id ProcessName

------- ------ ----- ----- ----- ------ -- -----------

258 8 29772 38636 130 3700 powershell

398 24 75988 76800 572 5816 powershell

605 9 30668 29800 155 7.11 3052 powershell

Имена компьютеров в этих данных не указаны, однако они хранятся в свойстве MachineName объектов процесса, возвращаемых Get-Process. Следующая команда использует командлет Format-Table для отображения свойств ID, ProcessName и MachineName (ComputerName) объектов процесса.

Единственным способом непосредственной блокировки компьютера с помощью стандартных средств является вызов функции **LockWorkstation()** в **user32.dll**:

rundll32.exe user32.dll,LockWorkStation

Эта команда немедленно блокирует рабочую станцию. Она использует **rundll32.exe**, который запускает библиотеки DLL Windows (и сохраняет их библиотеки для многократного использования), чтобы запустить user32.dll — библиотеку функций управления Windows.

Если рабочая станция блокируется при включенном быстром переключении пользователей, например в Windows XP, компьютер отображает экран входа в систему вместо того, чтобы запустить заставку текущего пользователя.

Чтобы завершить работу конкретных сеансов на сервере терминалов, используйте программу командной строки **tsshutdn.exe**.

**Выход из текущего сеанса**

Выйти из сеанса в локальной системе можно несколькими способами. Самый простой заключается в использовании программы командной строки удаленного рабочего стола или служб терминалов — **logoff.exe** (для получения дополнительных сведений введитеlogoff /? в командной строке PowerShell). Чтобы выйти из текущего активного сеанса, введите logoff без аргументов.

Можно также использовать средство **shutdown.exe** с параметром выхода:

shutdown.exe -l

Еще один вариант — использование инструментария WMI. Класс**Win32\_OperatingSystem** имеет метод **Shutdown**. Вызов метода с флагом 0 инициирует выход из системы:

Дополнительные сведения о методе **Shutdown** класса Win32\_OperatingSystem см. в [этой статье](https://docs.microsoft.com/ru-ru/windows/win32/cimwin32prov/shutdown-method-in-class-win32-operatingsystem).

Get-CimInstance -Classname Win32\_OperatingSystem | Invoke-CimMethod -MethodName Shutdown

**Завершение работы или перезапуск компьютера**

Завершение работы и перезапуск компьютеров обычно относятся к схожим типам задач. Средства, завершающие работу компьютера, обычно также перезапускают его и наоборот. Есть два варианта непосредственной перезагрузки компьютера из PowerShell. Используйте **tsshutdn.exe** или **shutdown.exe** с соответствующими аргументами. Подробные сведения об использовании можно получить, запустив tsshutdn.exe /? илиshutdown.exe /?.

Операции завершения работы и перезапуска можно также выполнять непосредственно из PowerShell.

Чтобы завершить работу компьютера, используйте команду Stop-Computer.

Stop-Computer

Чтобы перезапустить операционную систему, используйте команду Restart-Computer.

PowerShellКопировать

Restart-Computer

Чтобы выполнить немедленную перезагрузку компьютера, используйте параметр -Force.

PowerShellКопировать

Restart-Computer -Force

### Конвейер

**PowerShell** позволяет осуществлять обмен данными между командлетами с помощью конвейера. Например:

* GetService | SortObject -property Status — сортировка запущенных служб по статусу;
* “Hello World!” | Out-File C:\ps\test.txt — запись текста в файл.

Можно использовать несколько конвейеров. Например, следующий скрипт выводит список имён всех служб за исключением остановленных:

Get-Service | WHERE {$\_.status -eq “Running”} | SELECT displayname

Литература

1. **Попов А. В.**  Введение в Windows PowerShell. — СПб.: БХВ-Петербург, 2009. —

464 с.: ил. — (Системный администратор)