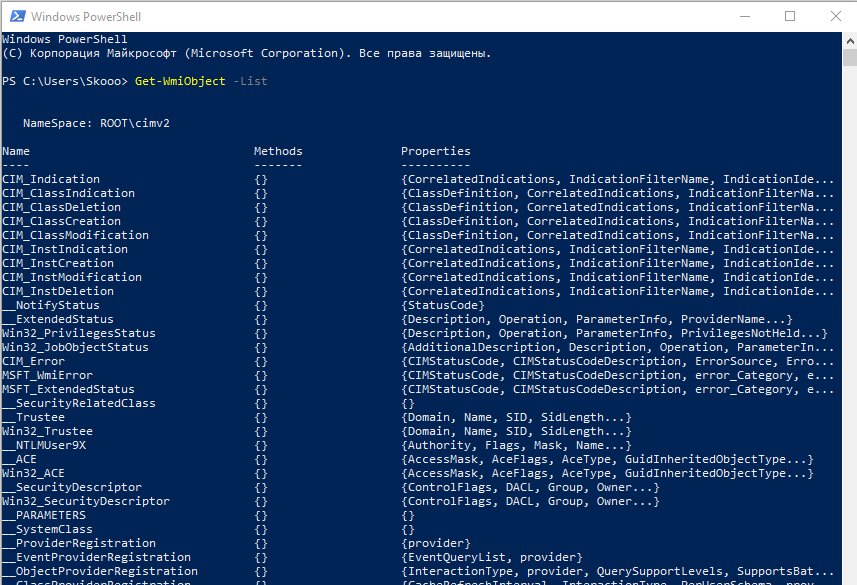
**Лабораторная работа 18**

**Работа в Windows PowerShell с объектными моделями WMI, COM и .NET**

**Подключение к подсистеме WMI. Получение списка классов**

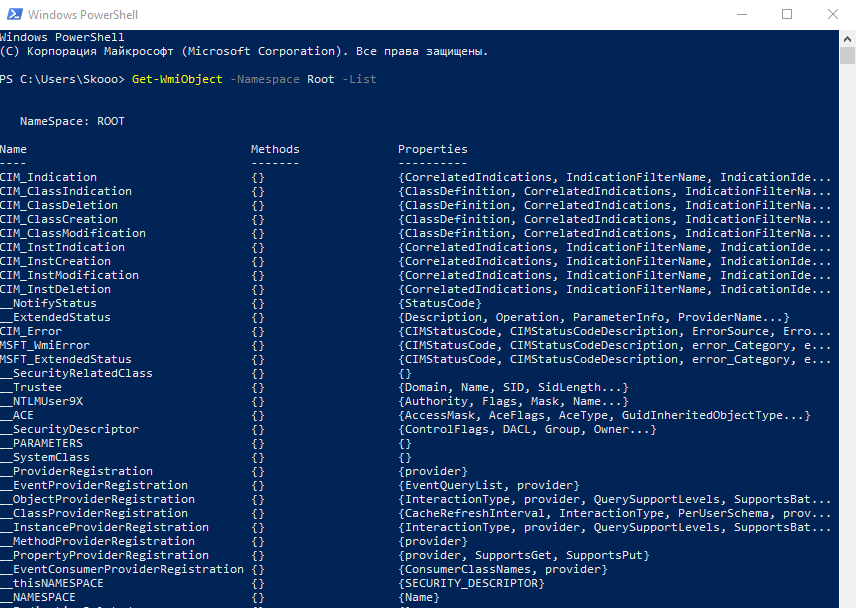
Для обращения к определенному объекту WMI нужно знать наименование класса, к которому он относится. PowerShell позволяет в интерактивном режиме получить список всех классов WMI на локальном или удаленном компьютере. Для этого нужно выполнить командлет Get-WmiObject с параметром List:

PS C:\> Get-WmiObject –List



По умолчанию командлет Get-WmiObject подключается к пространству имен Root\CIMV2. Сменить это пространство имен позволяет параметр Namespace, в качестве значения которого указывается нужное наименование. Например:

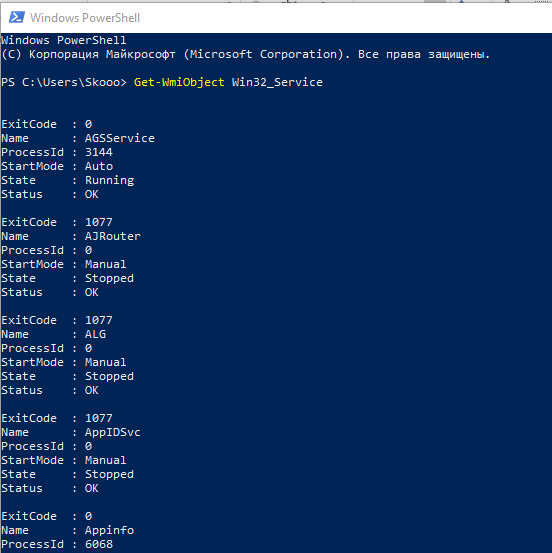
PS C:\> Get-WmiObject -Namespace Root -List



**Получение объектов WMI**

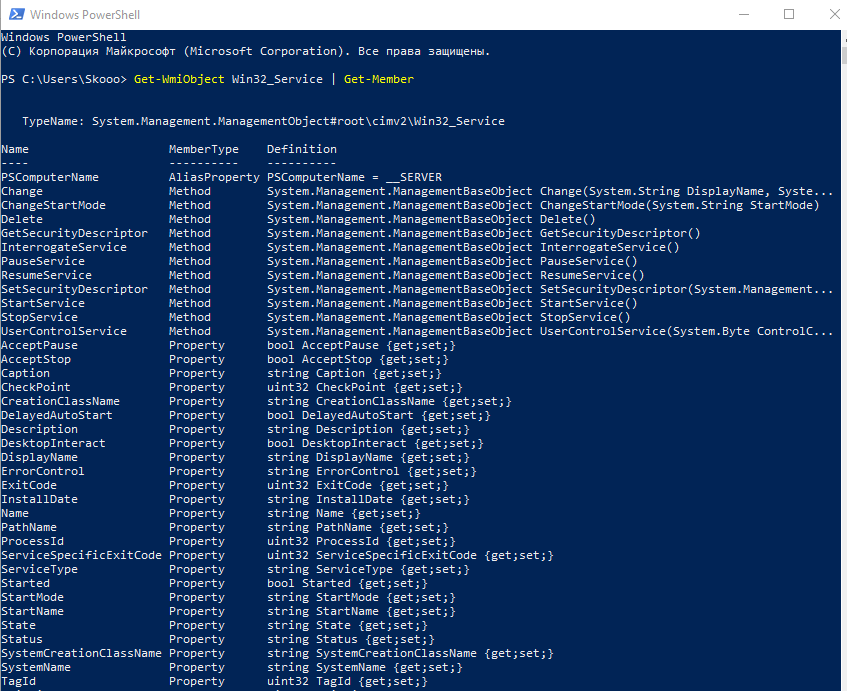
Зная имя класса WMI, получить экземпляры этого класса очень просто. Например:

PS C:\> Get-WmiObject Win32\_Service



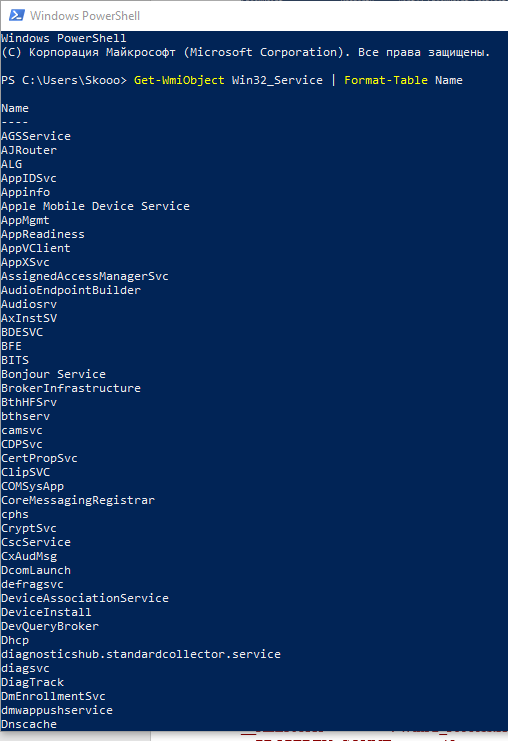
Данная команда выводит информацию о службах, зарегистрированных в системе. На самом деле объекты класса Win32\_Service имеют намного больше свойств, чем по умолчанию отображается на экране; увидеть список всех свойств и методов объекта WMI можно, как и в случае .NET-объектов, с помощью командлета Get-Member:

PS C:\> Get-WmiObject Win32\_Service | Get-Member



Используя командлеты форматирования, можно выводить на экран интересующие нас свойства, например:

PS C:\> Get-WmiObject Win32\_Service | Format-Table Name,

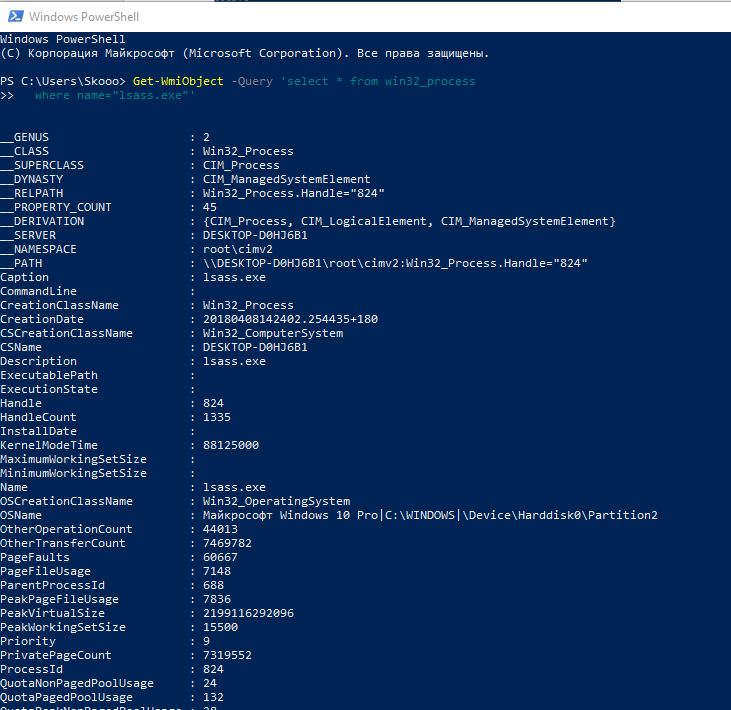


**Выполнение WQL-запросов**

Параметр Query командлета Get-WmiObject позволяет выполнять произвольные запросы на языке WQL. Например:

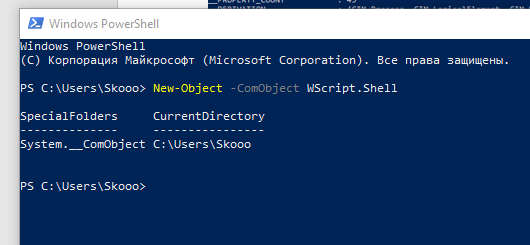
PS C:\> Get-WmiObject -Query 'select \* from win32\_process

where name="lsass.exe"'



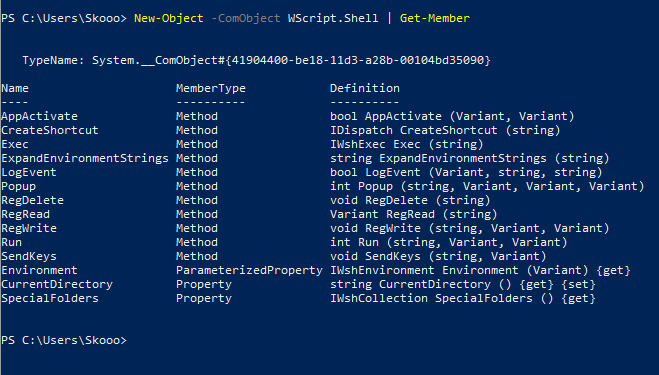
**Работа с COM-объектами**

PS C:\> New-Object -ComObject WScript.Shell



Перенаправив созданный объект по конвейеру на командлет Get-Member, можно увидеть список всех свойств и методов, имеющихся в COM-объекте:

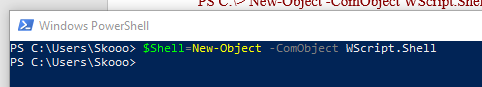
PS C:\> New-Object -ComObject WScript.Shell | Get-Member



**Сохранение объектов в переменных**

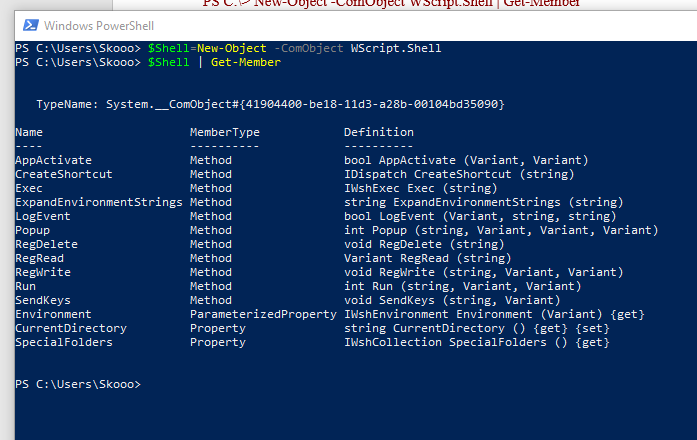
Для сохранения в переменной ссылки на COM-объект нужно выполнить следующую команду:

$Shell=New-Object -ComObject WScript.Shell



Убедимся, что в переменной $Shell на самом деле хранится экземпляр COM-объекта с программным идентификатором WScript.Shell:

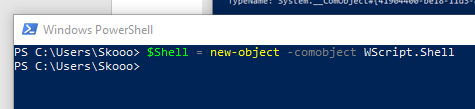
PS C:\> $Shell | Get-Member



**Пример: создание ярлыка на рабочем столе**

Сначала создадим экземпляр COM-объекта WScript.Shell и сохраним ссылку на него в переменной $Shell:

PS C:\> $Shell = new-object -comobject WScript.Shell



У данного объекта имеется метод CreateShortcut, в качестве параметра которого нужно указывать путь к создаваемому ярлыку. Путь к рабочему столу можно определить разными способами, например, с помощью переменной определенной в PowerShell переменной $Home, в которой хранится путь к личной папке активного пользователя:

PS C:\> $Home

C:\Documents and Settings\User



По умолчанию содержимое рабочего стола хранится в подкаталоге "Рабочий стол" личной папки пользователя, поэтому для создания ярлыка выполним следующую команду:

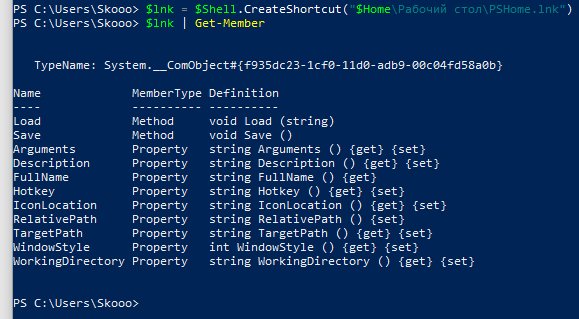
PS C:\> $lnk = $Shell.CreateShortcut("$Home\Рабочий стол\PSHome.lnk")



Здесь нужно учесть, что для получения значения переменной (в нашем случае $Home) ее имя нужно указывать внутри двойных кавычек, а не в одинарных.

Посмотрим, какие свойства и методы имеет объект, сохраненный в переменной $lnk:

PS C:\> $lnk | Get-Member



**Пример: вывод текста в Microsoft Word**

Сделать это несложно. Сначала создаем экземпляр главного объекта сервера автоматизации Microsoft Word, который имеет программный идентификатор "Word.Application", и сохраняем ссылку на этот объект в переменной $word::

PS C:\> $word = new-object -comobject Word.Application

Создаем новый документ (метод Add коллекции Documents ) и сохраняем ссылку на документ в переменной $doc:

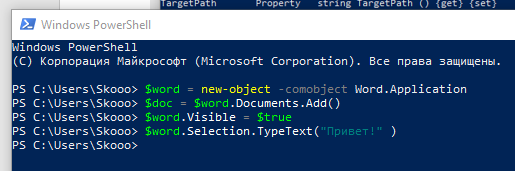
PS C:\> $doc = $word.Documents.Add()

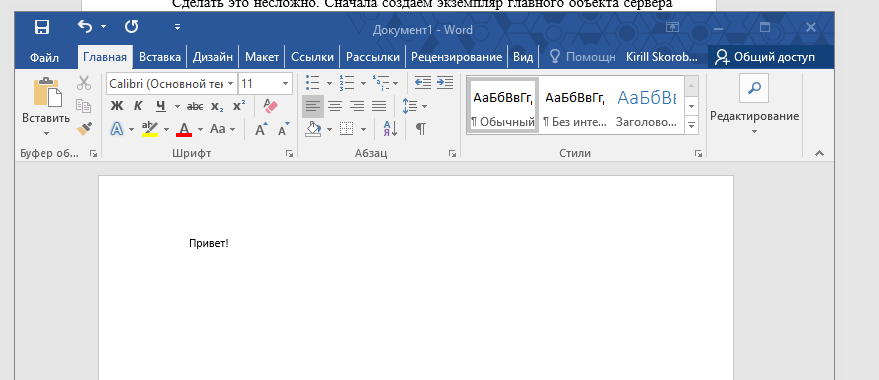
Делаем окно с новым документом видимым:

PS C:\> $word.Visible = $true

Наконец, печатаем строку текста:

PS C:\> $word.Selection.TypeText("Привет!" )



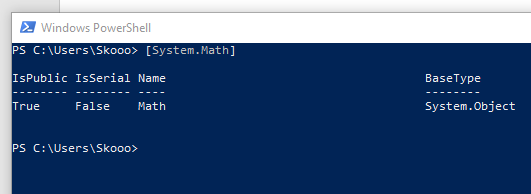


**Использование .NET-объектов. Вызов статических методов**

Также в .NET имеются классы, экземпляры которых нельзя создать с помощью командлета New-Object. Такие классы называются статическими, так как они не создаются, не уничтожаются и не меняются. В частности, статическим является класс System.Math, методы которого часто используются для математических вычислений.

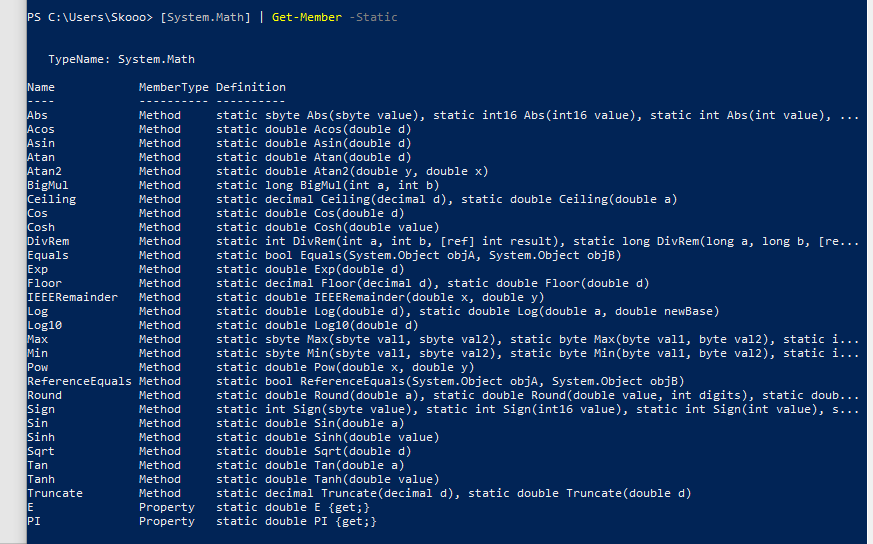
Для обращения к статическому классу его имя следует заключить в квадратные скобки, например:

PS C:\> [System.Math]



Методы статического класса также называются статическими. Для просмотра статических методов класса нужно передать имя этого класса (в квадратных скобках) по конвейеру командлету Get-Member с параметром Static:

PS C:\> [System.Math] | Get-Member –Static



Как мы видим, методы класса System.Math реализуют различные математические функции, их легко распознать по названию.

Для доступа к определенному статическому методу или свойству используются два идущих подряд двоеточия ( ::), а не точка ( .), как в обычных объектах. Например, для вычисления квадратного корня из числа (статического метода Sqrt ) и сохранения результата в переменную используется следующая конструкция:

PS C:\> $a=[System.Math]::Sqrt(25)

PS C:\> $a

5

