Переменные Powershell



newadmin.ru/peremennie-powershell

14 сентября, 2020



Powershell

= <u>admin</u>

Переменные Powershell объявляются при помощи знака \$ и могут быть созданы в любом месте скрипта. При создании переменной нет необходимости указывать ее тип, система попытается определить его сама. В имени переменной допустимы любые буквы, цифры и знак подчеркивания.

Создание переменных

Давайте создадим несколько новых переменных и проверим что все получилось

- 1 \$а="Переменная1"
- 2 \$b="12345"
- \$с=Переменная 3
- \$d=12345

```
Agmинистратор: Windows PowerShell

PS C:\WINDOWS\system32> $a="Переменная1"

>> $b="12345"

>> $c=Переменная 3

>> $d=12345

Переменная: Имя "Переменная" не распознано как имя командлета, функции, фай ла сценария или выполняемой программы. Пров ерьте правильность написания имени, а также наличие и правильность пути, пос ле чего повторите попытку. 

строка: 3 знак: 4

+ $c=Переменная 3

+ CategoryInfo : ObjectNotFound: (Переменная:String) [], Comman dNotFoundException

+ FullyQualifiedErrorId: CommandNotFoundException

PS C:\WINDOWS\system32> $a
Переменная1

PS C:\WINDOWS\system32> $b
12345

PS C:\WINDOWS\system32> $c

PS C:\WINDOWS\system32> $d
12345

PS C:\WINDOWS\system32> $d
12345

PS C:\WINDOWS\system32>
```

Создание переменных

Мы видим что успешно созданы 3 переменные: **\$a**, **\$b**, **\$d** . Переменная **\$c** не создалась и вылезла ошибка. Дело в том что в переменной **\$c** при объявлении мы добавили пробел но не поставили кавычки. Если необходимо занести в переменную текст с пробелом нужно обязательно использовать кавычки. Посмотрим тип созданных переменных.

```
1  $a.GetType().Fullname
2  $b.GetType().Fullname
3  $c.GetType().Fullname
4  $d.GetType().Fullname
```

Тип переменных

Переменная **\$a** и **\$b** имеют тип *строка* (**System.String**). Если с **\$a** все понятно, там использовались буквы и цифра, то **\$b** стала строкой только потому что мы занесли цифры в кавычки. А в переменной **\$d** мы использовали цифры без кавычек, поэтому тип переменной стал *число* (**System.Int32**). Переменная **\$c** выпала с ошибкой NULL потому как её не существует.

Важно помнить, если вы случайно попытаетесь присвоить значение переменной которая уже существует, старое значение в ней изменится на новое. Поэтому необходимо быть осторожным в именовании переменных, и перед созданием новой, просто введите в Powershell её название и нажмите *Enter*. Если переменная не существует то система ничего не выведет кроме пустой строки.

Хорошо, а если нам необходимо изменить тип переменной на нужный нам? Например сделать из переменной **\$d** строку. Это возможно:

```
1 $d=[string]$d
```

Мы просто указываем в квадратных скобках переменную **string** и на выходе получаем уже **System.String**

Типы данных Powershell

Рассмотрим какие типы данных бывают в Powershell

Тип	Класс .NET	Описание
[string]	System.String	Строка
[char]	System.Char	Символ (как элемент кода UTF-16)
[bool]	System.Boolean	Булево (принимает значение \$true или \$false)
[int]	System.Int32	32-разрядное целое число
[long]	System.Int64 64-разрядное целое число	
[decimal]	System.Decimal 128 битное десятичное число. конце числа буква d обязатель	
[single]	System.Single	Число одиночной точности с плавающей запятой
[double]	System.Double	Число двойной точности с плавающей запятой
[DateTime]	System.DateTime	Представляет текущее время выраженное датой и временем суток.
[array]	System.Object[]	Массив. Предоставляет методы для создания, изменения, поиска и сортировки массивов
[hashtable]	System.Collections.Hashtable	Хеш-таблицы. Представляет коллекцию пар «ключ-значение», которые упорядочены по хэш-коду ключа. Отличие от массивов это использование именованных ключей вместо индексов.

Типы данных Powershell

Давайте присвоим переменной \$е булево значение

1 \$e=\$True

Теперь **\$e** имеет тип **System.Boolean**, добавим кавычки и посмотрим что будет

1 \$e="\$True"

Переменная стала **System.String**. Сделаем из нее снова булево, но кавычки убирать не будем

1 \$e=[bool]\$e

```
Администратор: Windows PowerShell
                                              ×
PS C:\WINDOWS\system32> $e=$True
System.Boolean
PS C:\WINDOWS\system32> $e="$True"
PS C:\WINDOWS\system32> $e.GetType().Fullname
System.String
PS C:\WINDOWS\system32> [bool]$e
PS C:\WINDOWS\system32> $e.GetType().Fullname
System.String
PS C:\WINDOWS\system32> $e=[bool]$e
PS C:\WINDOWS\system32> $e.GetType().Fullname
System.Boolean
PS C:\WINDOWS\system32>
```

Работаем с логикой

С датой также можно работать сделав преобразование из строки. Давайте создадим переменную **\$data** и добавим текст "01/01/2001" это будет в формате строки

```
1 $data="01/01/2001"
```

И теперь можно преобразовать эту строку в формат даты и посмотреть на вывод

1 [datetime]\$data

```
➢ Администратор: Windows PowerShell — 
PS C:\WINDOWS\system32> $data="01/01/2001"
PS C:\WINDOWS\system32> $data
01/01/2001
PS C:\WINDOWS\system32> $data.GetType().Fullname
System.String
PS C:\WINDOWS\system32> [datetime]$data

1 января 2001 г. 0:00:00

PS C:\WINDOWS\system32>

V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32>
V

PS C:\WINDOWS\system32
V

PS C:\WINDOWS\system3
```

Преобразуем строку в дату

Тут уже и без **GetType** видно что Powerhell преобразовал строку в дату, даже время поставил:). Данное преобразование может быть удобно для работы с таблицами. Когда необходимо импортировать таблицу с данными и позже преобразовать их из строк в другие типы данных.

Массив имеет тип **System.Object[]** и каждый элемент в массиве может иметь свой тип. Создадим переменную **\$massiv** и добавим в неё 3 значения: два строковых и одно число. Затем определим тип каждого элемента обратившись к нему по очереди. Обратиться к элементу массива можно указав его порядковый номер в скобках **П**. Расчет начинается с **0**.

1 \$massiv="element1",2,"tri"

```
Администратор: Windows PowerShell
                                                                   X
PS C:\WINDOWS\system32> $massiv="element1",2,"tri
PS C:\WINDOWS\system32> $massiv
element1
tri
PS C:\WINDOWS\system32> $massiv.GetType()
IsPublic IsSerial Name
                                                            BaseType
True
                  Object[]
         True
                                                            System.Array
PS C:\WINDOWS\system32> $massiv.GetType().Fullname
System.Object[]
PS C:\WINDOWS\system32> $massiv[0].GetType().Fullname
System.String
PS C:\WINDOWS\system32> $massiv[1].GetType().Fullname
System.Int32
PS C:\WINDOWS\system32> $massiv[2].GetType().Fullname
System.String
PS C:\WINDOWS\system32>
```

Работа с массивом

Рассмотрим последний тип данных это **hashtable**. Хеш-таблицы отличаются от массива тем что вместо индекса используют пару "ключ-значение". Объявляется в таком формате: @{kluch1="Nomer1";kluch2="Nomer2";kluch3="Nomer3"}. Давайте создадим переменную \$hash и занесем в нее хеш-таблицу

1 \$hash=@{kluch1="Nomer1";kluch2="Nomer2";kluch3="Nomer3"}

```
Aдминистратор: Windows PowerShell — X
PS C:\WINDOWS\system32> $hash=@{kluch1="Nomer1";kluch2="Nomer2";kluch3="Nomer3"}
PS C:\WINDOWS\system32> $hash

Name Value
----
kluch2 Nomer2
kluch1 Nomer1
kluch3 Nomer3

PS C:\WINDOWS\system32>
```

Хеш-таблица

Тут как в массиве, каждый элемент может иметь свой тип. Можно применять всевозможные сортировки, выборки и исключения.

Область действия

Необходимо помнить, все созданные пользователем переменные хранятся только в текущем сеансе Powershell. После закрытия окна консоли и повторного открытия переменных уже не будет.

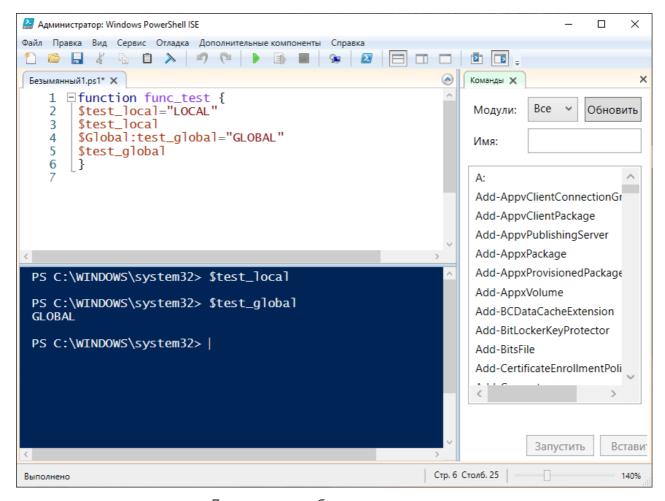
Существует две области действия переменных: **глобальная** и **локальная**. Глобальная область действия используется на весь ceaнс Powershell. Она состоит из переменных Powershell, системных переменных и переменных определенных в консоли (в рамках текущего ceaнса Powershell). Локальные переменные работаю только в той области для которой они определены (функция или скрипт) не выходя за ее пределы.

Рассмотрим работу глобальной и локальной переменной на небольшом примере:

```
1 function func_test {
2  $test_local="LOCAL"
3  $test_local
4  $Global:test_global="GLOBAL"
5  $test_global
6 }
```

Мы создали функцию в которой прописали две переменные **\$test_local** – локальная переменная, действует только внутри этой функции. Переменная **\$test_global** – глобальная, данные в ней сохраняются даже после завершения работы функции, в

отличие от локальной.

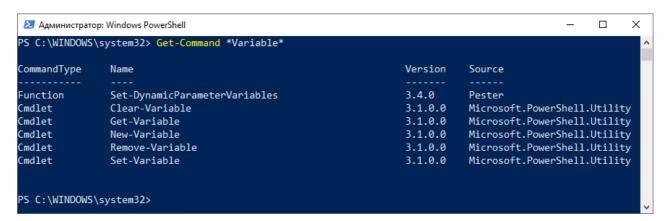


Локальная и глобальная переменные

Работа с переменными

Для работы с переменными существует 5 командлетов. Давайте посмотрим какие

1 Get-Command *Variable*



Get-Command *Variable*

Для удобства сведем по ним данные в таблицу

Название	Назначение	Описание
Get- Variable	Получить значение переменой	Позволяет получить значение переменной со всеми дополнительными параметрами. Если не указать название переменной, командлет выведет список всех текущих переменных Powershell.
New- Variable	Создать новую переменную	Создает новую переменную с возможностью задать дополнительные параметры. Можно создать переменную только для чтения, скрытую переменную или переменную с пробелом в имени
Set- Variable	Изменить переменную	Изменение значений в переменной. В случае отсутствия переменной с указанным именем, она будет создана.
Clear- Variable	Очистить переменную	Удаляет все значения в переменной, но не саму переменную. Тип переменой остается тот же что был до очистки значений.
Remove- Variable	Удалить переменную	Удаляет переменную со всеми ее значениями

Работа с переменными

Запустим командлет Get-Variable

1 Get-Variable

```
Администратор: Windows PowerShell
                                                                                                       PS C:\WINDOWS\system32> Get-Variable
Name
                                cls
                                True
                                cls
args
ConfirmPreference
                                High
ConsoleFileName
DebugPreference
                                SilentlyContinue
Error
                                {Не удается удалить переменную rd, так как она является постоянной либо ...
ErrorActionPreference
                                Continue
                                NormalView
ExecutionContext
                                System.Management.Automation.EngineIntrinsics
                               False
false
FormatEnumerationLimit
                                {kluch2, kluch1, kluch3}
hash
HOME
                                C:\Users\Arthur
Host
                                System.Management.Automation.Internal.Host.InternalHost
InformationPreference
                                SilentlyContinue
input
                                System.Collections.ArrayList+ArrayListEnumeratorSimple
massiv
                                {element1, 2, tri}
MaximumAliasCount
                                4096
MaximumDriveCount
                                4096
MaximumErrorCount
                                256
MaximumFunctionCount
                                4096
MaximumHistoryCount
                                4096
MaximumVariableCount
MyInvocation
                                System.Management.Automation.InvocationInfo
NestedPromptLevel
null
OutputEncoding
                                System.Text.ASCIIEncoding
PID
                               4764
PROFILE
                                C:\Users\Arthur\Documents\WindowsPowerShell\Microsoft.PowerShell_profile...
ProgressPreference
                               Continue
PSBoundParameters
                                {}
PSCommandPath
PSCulture
                                ru-RU
PSDefaultParameterValues
                                {}
PSEdition
                                Desktop
PSEmailServer
PSHOME
                                C:\Windows\System32\WindowsPowerShell\v1.0
PSScriptRoot
PSSessionApplicationName
                                wsman
PSSessionConfigurationName
                                http://schemas.microsoft.com/powershell/Microsoft.PowerShell
PSSessionOption
                                System.Management.Automation.Remoting.PSSessionOption
PSUICulture
                                ru-RU
                                {PSVersion, PSEdition, PSCompatibleVersions, BuildVersion...}
PSVersionTable
PWD
                                C:\WINDOWS\system32
ShellId
                                Microsoft.PowerShell
StackTrace
                                  в System.Runtime.ExceptionServices.ExceptionDispatchInfo.Throw()...
true
                                True
VerbosePreference
                                SilentlyContinue
```

Get-Variable

Мы видим список всех переменных Powershell, большая часть из них это **системные переменные**. Давайте посмотрим параметры переменной hash, та что чуть выше мы создали для хеш-таблицы.

1 Get-Variable hash

```
Администратор: Windows PowerShell
                                                                                                PS C:\WINDOWS\system32> Get-Variable hash|Format-List
PSPath
             : Microsoft.PowerShell.Core\Variable::hash
PSI-c
             : Microsoft.PowerShell.Core\Variable
PSIsContainer : False
Name
             : hash
Description :
             : {kluch2, kluch1, kluch3}
Value
Visibility
             : Public
Module
ModuleName
Options
             : None
Attributes
             : {}
PS C:\WINDOWS\system32>
```

Get-Variable hash

Видим что тут большее число параметров чем можно указать при заведение через знак \$. Попробуем создать новую переменную только для чтения. После ее создания значение переменной нельзя будет изменить.

```
1 New-Variable -Name rd -Option ReadOnly -Value 100 -Description "Только для чтения"
2
Get-Variable -Name rd|Format-List
```

```
Администратор: Windows PowerShell
                                                                                                     ×
PS C:\WINDOWS\system32> New-Variable -Name rd -Option ReadOnly -Value 100 -Description
PS C:\WINDOWS\system32> Get-Variable rd|Format-List
Name
           : rd
Description : Только для чтения
          : 100
Value
Visibility : Public
Module
ModuleName
Options 0
           : ReadOnly
Attributes : {}
PS C:\WINDOWS\system32> Get-Variable rd|Format-List
```

New-Variable rd

Попробуем изменить в ней значение

```
1 Set-Variable - Name rd - Value 200
```

```
Aдминистратор: Windows PowerShell

PS C:\WINDOWS\system32> Set-Variable rd -Value 200
Set-Variable : Не удается перезаписать переменную rd, так как она является постоянной либо доступна только для чтения.

строка:1 знак:1
+ Set-Variable rd -Value 200
+ CategoryInfo : WriteError: (rd:String) [Set-Variable], SessionStateUnauthorizedAccessException + FullyQualifiedErrorId : VariableNotWritable,Microsoft.PowerShell.Commands.SetVariableCommand

PS C:\WINDOWS\system32>
```

Set-Variable rd

Появилась ошибка, потому что переменная только для чтения. Данный параметр бывает полезен в скриптах. Его используют для постоянной переменной которую нельзя менять. Теперь удалим нашу переменную

1 Remove-Variable -Name rd -Force

Параметр **-Froce** указывать обязательно, иначе переменную только для чтения не удалить.

Удобство использования переменных заключается в том, что мы можем использовать ее для сокращения записи. Например, получим список процессов firefox с параметрами используемой памяти

1 \$fire=(Get-Process -Name firefox)|select Name, PagedMemorySize64

```
Администратор: Windows PowerShell
PS C:\WINDOWS\system32> $fire=(Get-Process -Name firefox)|select Name, PagedMemorySize64
PS C:\WINDOWS\system32> $fire
        PagedMemorySize64
Name
firefox
                33988608
firefox
                234557440
                194920448
firefox
                102944768
firefox
firefox
                 47992832
firefox
                 87834624
firefox
                 87478272
firefox
                 78905344
PS C:\WINDOWS\system32>
```

\$fire

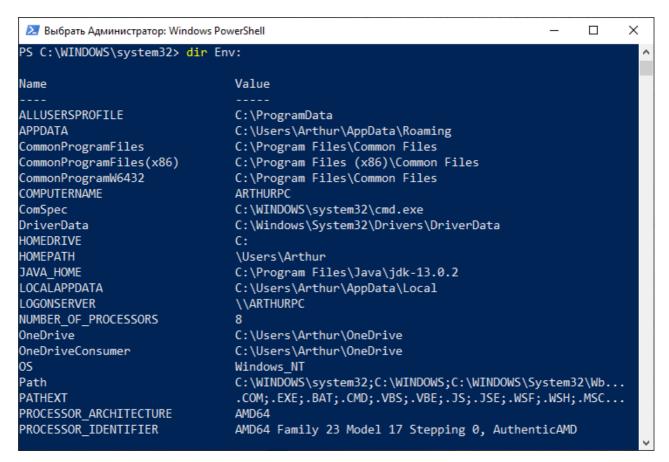
Мы можем продолжать дальше работать с этой переменной, можем выделить первый ее элемент

1 \$fire|Select-Object -First 1

Переменные окружения

Переменные окружения Windows также доступны из Powershell. Доступ к ним можно получить через диск **ENV**:

1 dir ENV:



dir env:

Рекомендую к прочтению:

- Powershell скрипты
- Операторы сравнения
- Операторы условий
- Циклы