

Операторы сравнения Powershell

 newadmin.ru/operatory-sravneniya-powershell

16 сентября, 2020



Powershell

 admin

В данной статье мы рассмотрим не только операторы сравнения Powershell, но и операторы замены, логические операторы. Сравнивать можно не только данные с типом строка или число, но и дата, булево и т.п.

В Powershell используются операторы сравнения на основе букв, что поначалу не привычно. Очень хочется написать знак `>` вместо `-gt` или знак `=` вместо `-eq`. Но со временем к такому именованию привыкаешь. Перед оператором сравнения или логическим оператором обязательно нужно ставить знак `-`

Операторы сравнения

Оператор	Значение	Описание
-eq	Equal	Равно
-ne	Not equal	Не равно
-gt	Greater than	Больше
-ge	Greater than or equal	Больше или равно
-lt	Less than	Меньше
-le	Less than or equal	Меньше или равно
-like	Wildcard comparison	Используется подстановка символов для поиска по шаблону
-notlike	Wildcard comparison	Используется подстановка символов для поиска не соответствия указанному шаблону
-match	Regular expression comparison	Использование регулярных выражений для поиска по шаблону
-notmatch	Regular expression comparison	Использование регулярных выражений для поиска не соответствия указанному шаблону
-contains	Containment operator	Определяет, содержит ли значение слева от оператора значение справа
-notcontains	Containment operator	Определяет, что значение слева от оператора не содержит значение оператора справа
-in	Containment operator	Возвращает true если искомое значение содержится в коллекции
-notin	Containment operator	Возвращает true если искомое значение не содержится в коллекции
-replace	Replace operator	Заменяет часть или все значение слева от оператора
-is	Type	Возвращает true если оба объекта одинаковы
-isnot	Type	Возвращает true если оба объекта разные

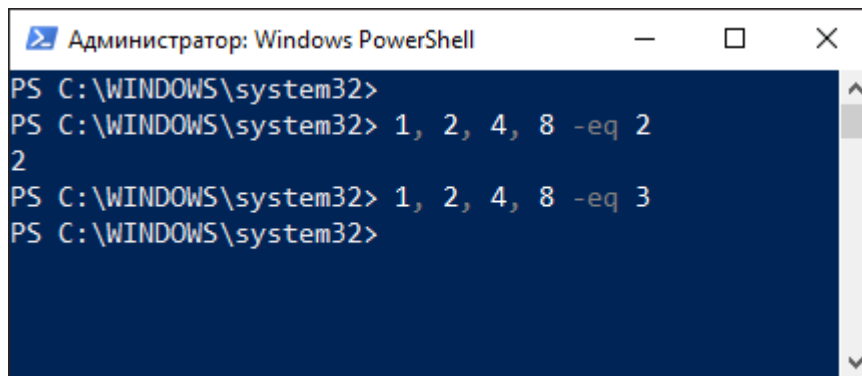
По умолчанию все операторы сравнения нечувствительны к регистру. Чтобы сделать оператор сравнения чувствительным к регистру, перед именем оператора поставьте букву **s**. К примеру, чувствительной к регистру оператор **-eq** пишется как **-se**

seq. Чтобы сделать нечувствительность к регистру явной, перед оператором ставится **i**. К примеру, явно нечувствительная к регистру версия оператора **-eq** это **-ieq**.

Когда входные данные для оператора являются скалярным значением (число), операторы сравнения возвращают логическое значение. Если входные данные представляют собой набор значений, операторы сравнения возвращают любые совпадающие значения. Если в коллекции нет совпадений, операторы сравнения возвращают пустой массив. Исключение составляют операторы **contains**, **notcontains**, **in**, **notin**, **is**, **isnot** которые всегда возвращают **логическое** значение.

1 1, 2, 4, 8 -eq 2

2 1, 2, 4, 8 -eq 3



```
Администратор: Windows PowerShell
PS C:\WINDOWS\system32>
PS C:\WINDOWS\system32> 1, 2, 4, 8 -eq 2
2
PS C:\WINDOWS\system32> 1, 2, 4, 8 -eq 3
PS C:\WINDOWS\system32>
```

В первом варианте при сравнении набора чисел с 2 такое число есть в массиве. Во втором варианте при сравнении с 3 возвращается пустой массив, такого числа нет.

Операторы равенства

Операторы равенства (-eq, -ne) возвращают значение **TRUE** или совпадения, когда одно или несколько входных значений идентичны указанному шаблону. Шаблон должен полностью соответствовать значению.

Рассмотрим несколько простых примеров

1 20 -eq 20

2 21 -eq 20

3 "Проверка" -eq "Проверка"

4 "Проверка" -eq "Проверка не прошла"

5 "Test_ne" -ne "Test"

```
Администратор: Windows PowerShell
PS C:\WINDOWS\system32> 20 -eq 20
True
PS C:\WINDOWS\system32> 21 -eq 20
False
PS C:\WINDOWS\system32> "Проверка" -eq "Проверка"
True
PS C:\WINDOWS\system32> "Проверка" -eq "Проверка не  
прошла"
False
PS C:\WINDOWS\system32> "Test_ne" -ne "Test"
True
PS C:\WINDOWS\system32>
```

Сравним между собой две даты.

- 1 (Get-Date) -eq [datetime]("09.15.2020")
- 2 (Get-Date).Date -eq [datetime]("09.15.2020")

```
Администратор: Windows PowerShell
PS C:\> Get-Date

15 сентября 2020 г. 21:27:27

PS C:\> (Get-Date).Date

15 сентября 2020 г. 0:00:00

PS C:\> (Get-Date) -eq [datetime]("09.15.2020")
False
PS C:\> (Get-Date).Date -eq [datetime]("09.15.2020")
True
PS C:\>
```

Давайте посмотрим на окошко вывода и разберемся что произошло. В первой строке я воспользовался командлетом Get-Date для получения текущей даты и времени. И сравнил его со строкой преобразованной в дату. Однако результат сравнения отрицательный т.е. **False**. Как видно число полностью совпадает, но, не

совпадает время. Поэтому я взял командлет `Get-Date` и вывел из него только текущую дату (в таком случае время устанавливается в 0). И уже второй строкой результат сравнения положительный.

Давайте посмотрим на следующий пример который наверняка пригодится в хозяйстве. Мы будем искать файлы больше определенного размера.

```
1 Get-ChildItem -Path c:\ -Recurse|Where-Object {$_.Length -ge 40000000}
```

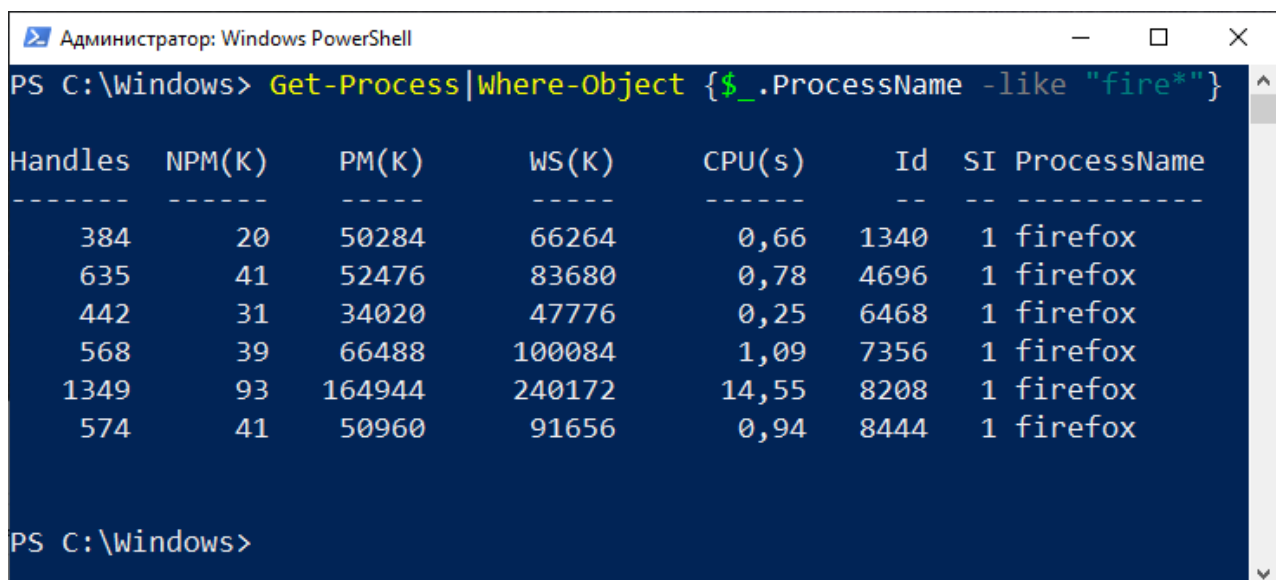
Итак, командлетом **Get-ChildItem** я хочу вывести список всех файлов и папок на диске C включая подпапки. Далее отправляя по конвейеру командлету `Where-Object` я фильтрую вывод, чтобы показывались только файлы больше или равные (оператор **-ge**) 40 Мбайт. Как видите очень удобный инструмент получается. Особенно в качестве автоматизации чистки логов.

Операторы сопоставления

Операторы сопоставления (**-like** и **-notlike**) находят элементы, которые соответствуют или не соответствуют заданному шаблону, используя подстановочные выражения.

Рассмотрим на примере списка запущенных процессов. Получим список запущенных процессов **firefox**

```
1 Get-Process|Where-Object {$_.ProcessName -like "fire*"}
```



Администратор: Windows PowerShell

```
PS C:\Windows> Get-Process|Where-Object {$_.ProcessName -like "fire*"}
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id	SI	ProcessName
384	20	50284	66264	0,66	1340	1	firefox
635	41	52476	83680	0,78	4696	1	firefox
442	31	34020	47776	0,25	6468	1	firefox
568	39	66488	100084	1,09	7356	1	firefox
1349	93	164944	240172	14,55	8208	1	firefox
574	41	50960	91656	0,94	8444	1	firefox

```
PS C:\Windows>
```

Командлетом **Where-Object** я выбрал из списка процессов (при помощи оператора **-like**) все с названием **fire** и дальше любые символы (*-любое кол-во символов)

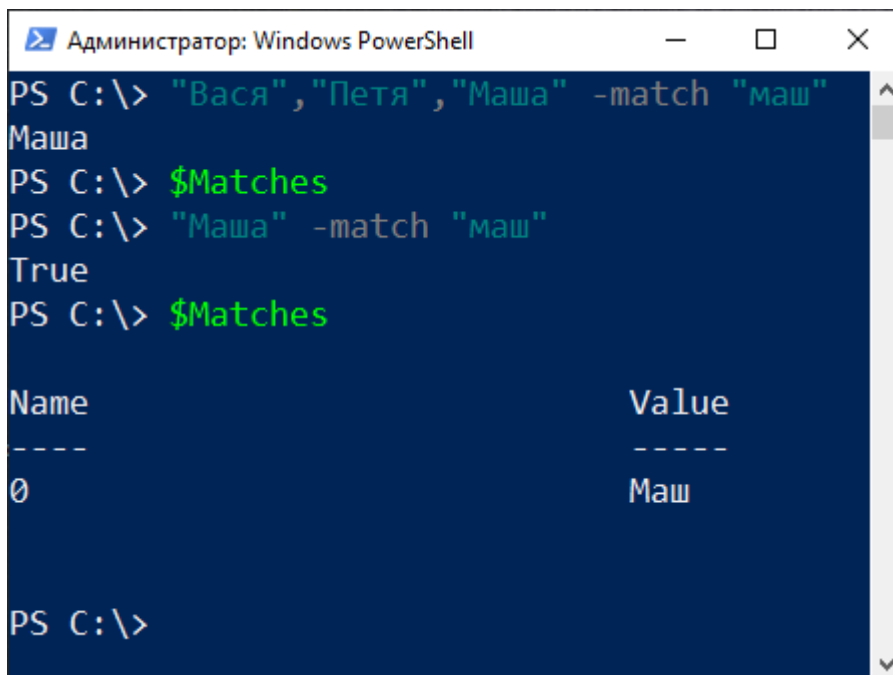
Очень удобная конструкция. Всегда её использую для поиска необходимых процессов.

Операторы соответствия (*-match* и *-notmatch*) находят элементы, которые соответствуют или не соответствуют заданному шаблону, используя регулярные выражения.

Оператор сопоставляет строку с помощью регулярных выражений. Если данные на входе являются скалярными, они заполняют автоматическую переменную **\$Matches**. Если входными данными является коллекция, операторы *-match* и *-notmatch* возвращают соответствующие члены этой коллекции, но при этом, оператор не заполняет переменную **\$Matches**.

Перейдем к примерам

- 1 "Вася", "Петя", "Маша" -match "маш"
- 2 \$Matches
- 3 "Маша" -match "маш"
- 4 \$Matches



```
Администратор: Windows PowerShell
PS C:\> "Вася", "Петя", "Маша" -match "маш"
Маша
PS C:\> $Matches
PS C:\> "Маша" -match "маш"
True
PS C:\> $Matches

Name                           Value
----                           -
0                               Маш
```

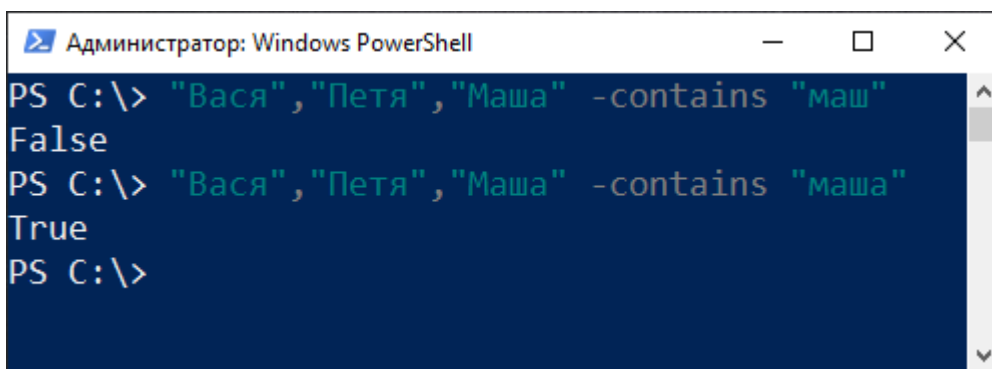
В случае сравнения с одной строкой *-match* возвращает логическое значение и заполняет автоматическую переменную **\$Matches**. Автоматическая переменная **\$Matches** – это хэш-таблица. Если группировка или захват не используются, заполняется только один ключ. Клавиша 0 представляет весь текст, который был сопоставлен.

Операторы сдерживания (-contains и -notcontains) аналогичны операторам равенства. Только операторы сдерживания всегда возвращают логическое значение, даже если входными данными является коллекция.

В отличие от операторов равенства, операторы сдерживания возвращают значение, как только они обнаруживают первое совпадение. Операторы равенства оценивают все входные данные, а затем возвращают все совпадения в коллекции.

Посмотрим на практике

- 1 "Вася", "Петя", "Маша" -contains "маш"
- 2 "Вася", "Петя", "Маша" -contains "маша"



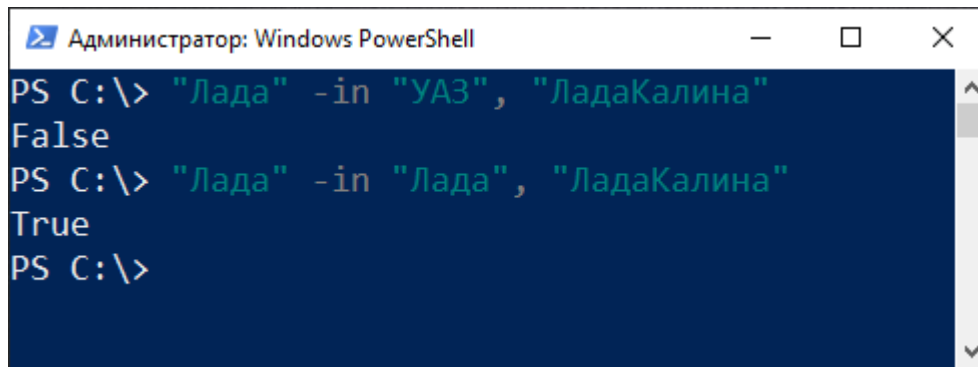
```
Администратор: Windows PowerShell
PS C:\> "Вася", "Петя", "Маша" -contains "маш"
False
PS C:\> "Вася", "Петя", "Маша" -contains "маша"
True
PS C:\>
```

Как видно из примера, в случае не полного совпадения имени результат **False**. Если имя совпадает результат **True**.

Оператор -in указывает, появляется ли тестовое значение в коллекции эталонных значений. Всегда возвращает логическое значение. Возвращает **TRUE** только в том случае, если тестовое значение точно соответствует хотя бы одному из эталонных значений. Когда тестовое значение является коллекцией, оператор **-in** использует ссылочное равенство. Он возвращает **TRUE** только в том случае, если одно из опорных значений является тем же экземпляром объекта. Оператор **-in** был введен в **PowerShell 3.0**

Примеры

- 1 "Лада" -in "УАЗ", "ЛадаКалина"
- 2 "Лада" -in "Лада", "ЛадаКалина"



```
Администратор: Windows PowerShell
PS C:\> "Лада" -in "УАЗ", "ЛадаКалина"
False
PS C:\> "Лада" -in "Лада", "ЛадаКалина"
True
PS C:\>
```

Оператор замены

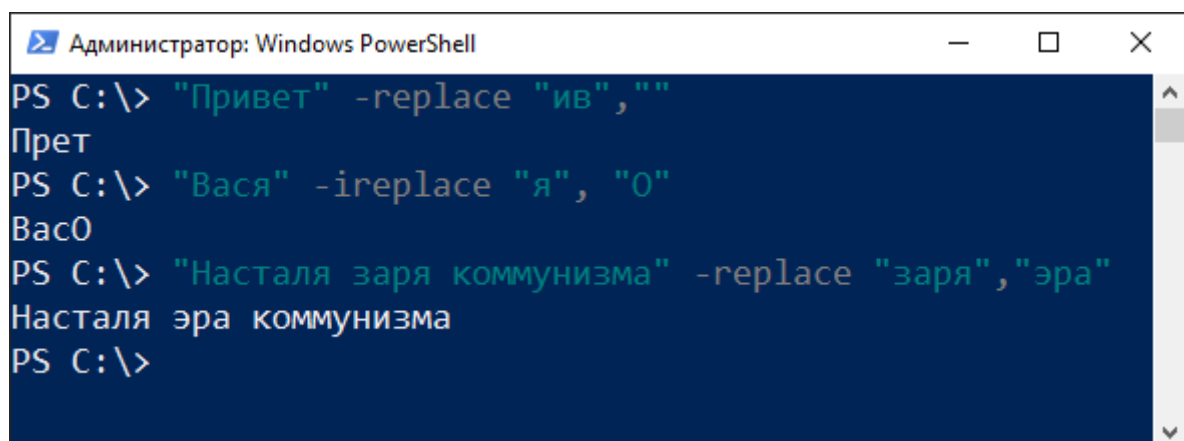
Оператор замены (***-replace***) заменяет все или часть значения указанным значением с помощью регулярных выражений. Оператор ***-replace*** можно использовать для многих административных задач, таких как переименование файлов, замена текста в файле и прочее.

Рассмотрим пример по массовому переименованию файлов *.log в *.bak

```
1 Get-ChildItem *.log | Rename-Item -NewName { $_.name -replace  
   '\.log$', '.bak' }
```

И немного простых примеров с текстом

```
1 "Привет" -replace "ив", ""
Прет
2 "Вася" -ireplace "я", "0"
Вас0
3 "Насталя заря коммунизма" -replace "заря", "эра"
Насталя эра коммунизма
PS C:\>
```



```
Администратор: Windows PowerShell
PS C:\> "Привет" -replace "ив", ""
Прет
PS C:\> "Вася" -ireplace "я", "0"
Вас0
PS C:\> "Насталя заря коммунизма" -replace "заря", "эра"
Насталя эра коммунизма
PS C:\>
```

Мы закончили рассматривать операторы сравнения Powershell. Теперь рассмотрим логические операторы.

Логические операторы Powershell

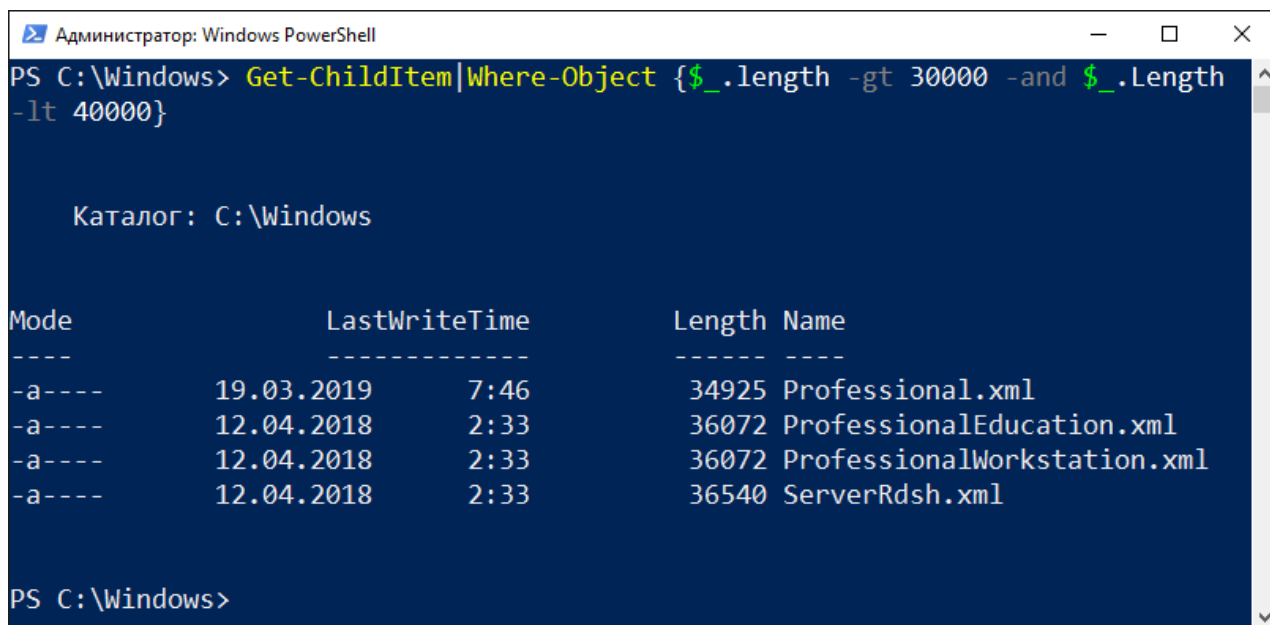
Оператор Описание

-and	Оба условия должны быть истины, чтобы выражение было истинно
-or	Одно или оба условия должны быть истины, чтобы выражение было истинно
-xor	Одно условие должно быть истинно, а второе должно иметь значение ложь, чтобы выражение было истинно
-not	Указанные условия должны иметь значение ложь, чтобы выражение было истинно
!	Указанное условие должно иметь значение ложь, чтобы выражение было истинно

Логические операторы PowerShell соединяют выражения и операторы, позволяя использовать одно выражение для проверки нескольких условий.

Снова обратимся к примеру из жизни. Появилась у меня задача найти файлы определенного размера. А именно чтобы они были больше одного числа но меньше другого. Реализуем на Powershell

```
1 Get-ChildItem|Where-Object {$_.length -gt 30000 -and $_.Length -lt 40000}
```



```
Администратор: Windows PowerShell
PS C:\Windows> Get-ChildItem|Where-Object {$_.length -gt 30000 -and $_.Length -lt 40000}

Каталог: C:\Windows

Mode                LastWriteTime         Length Name
----                -
-a----          19.03.2019         7:46         34925 Professional.xml
-a----          12.04.2018         2:33         36072 ProfessionalEducation.xml
-a----          12.04.2018         2:33         36072 ProfessionalWorkstation.xml
-a----          12.04.2018         2:33         36540 ServerRdsh.xml

PS C:\Windows>
```

В данном примере мы получили список файлов в папке Windows объемом больше 30 Кбайт но меньше 40 Кбайт.

Мы рассмотрели все операторы сравнения Powershell. Их более чем достаточно для любых задач.

Рекомендую к прочтению:

- Powershell скрипты
- Переменные
- Операторы условий
- Циклы