Kenneth R N Wilson

PowerShell

Cheat Sheets

Environment Variables and Terminal

Effect	Example
Version of PowerShell	`$PSVersionTable`
Enable local scripts unsigned	`Set-ExecutionPolicy RemoteSigned`
Get latest help on cmdlet	`Get-Help -Online dir`
Get aliases cmdlet	`Get-Command dir`
Get info on cmdlet truncated	`Get-Command Get=ChildItem`
Get all info on cmdlet	`Get-Command Get-ChildItem | Format-List`

Basics

Effect	Example
Expression	`(1+1) * 3`
Variable	`$r = 2 * 3`
String Concat	`"Hello" + " " + "World"`
Arrays	`$a = 1,2,3,4`

Command List

Effect	Command	Alias
List Directory Contents	`Get-ChildItem`	`dir`
Sort	`Sort-Object`	`sort`
Select subrange of objects	`Select-Object`	
Select properties on Objects	`Select-Object`	
For-Each	`ForEach-Object`	

Quoting

Effect	Example	Result
No Expansion	`$myvar = 4;` `Write-Output '$myvar'`	`$myvar`
Expansion	`$myvar = 4;` `Write-Output "$myvar"`	`4`
Escape double quote	`Write-Output "`"Hello`""`	`"Hello"`
Arrays	`$a = 1,2,3,4`	
Quoting basics	`'-Input'`	

Types

HashTable

```
# Create HashTable
$person = @{Name="Kenny";Age=24;};

# Display Contents
$person

# Index Using Dot Notation
$person.Name

# Index Using Array Notation
$person["Age"]
$person["Age", "Name"]

# Display Keys
$person.Keys

# Display Values
$person.Values

# Display Values
$person[$person.Keys]

# Display Keys Sorted
$person.Keys | Sort-Object -Descending

# Interate in foreach
foreach($kv in $person.GetEnumerator())
{
     $kv.key + ":" + $kv.value
}
```

Commands

Parameters and Arguments

Commands specify parameters and arguments. Consider the following.

```
command -parameter1 -parameter2 argument1 argument2
```

- -parameter1: Flag takes no argument
- -parameter2: Parameter that takes argument
- argument1: argument for -parameter2
- argument2: positional argument

Since arguments on the command line are strings and parameters can require objects of complex types, type conversion is used.

End Of Parameters

The special `--` is used to indicate this is the last named parameter.

Type of Commands

PowerShell has four kinds of commands.

Type	Descrption
Cmdlet	Backed by .NET type and always have the form Verb-Noun e.g., `Get-ChildItem`
Shell Function	Named piece of shell script logic.
Script	
Native Commands	

Pipeline

Working With

We start with three files in a directory. The following is the result of `dir`.

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        15/03/2021     16:48             12 FileOne.txt
-a----        15/03/2021     16:48            124 LongFile.txt
-a----        15/03/2021     16:47            187 MediumFile.txt
```

Filtering

We can filter the results using the `Where-Object` cmdlet.

```
dir | Where-Object -Property Length -GT 15
```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        15/03/2021     16:48            124 LongFile.txt
-a----        15/03/2021     16:47            187 MediumFile.txt
```

Sorting

We can sort by length descending order

```
dir | Sort-Object -Property Length -Descending
```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        15/03/2021     16:47            187 MediumFile.txt
-a----        15/03/2021     16:48            124 LongFile.txt
-a----        15/03/2021     16:48             12 FileOne.txt
```

Sub Range

We can use Select-Object to select a subrange.

```
dir | Sort-Object -Property Length -Descending | Select-Object -First 1
```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        15/03/2021     16:47            187 MediumFile.txt
```

Select Properties

We can select the output properties using `Select-Object`

```
dir | Sort-Object -Property Length -Descending | Select-Object -Property
Name, Length
```

```
Name           Length
----           ------
MediumFile.txt    187
LongFile.txt      124
FileOne.txt        12
```

Formatting

The actual information displayed depends on the type of the object. PowerShell comes with a set of installed configuration files that specify how different types of objects are displayed.

```
> dir $PSHOME/*format* | Format-Table name
```

```
Name
----
Certificate.format.ps1xml
Diagnostics.Format.ps1xml
DotNetTypes.format.ps1xml
Event.Format.ps1xml
FileSystem.format.ps1xml
Help.format.ps1xml
HelpV3.format.ps1xml
PowerShellCore.format.ps1xml
PowerShellTrace.format.ps1xml
Registry.format.ps1xml
WSMan.Format.ps1xml
```

Although we, in general, do not have control of these, we do have control of the shape of the output by choosing Format-* commands.

```
Get-Command Format-* | Format-Table name
```

```
Name
----
Format-Hex
Format-Volume
Format-Custom
Format-List
Format-SecureBootUEFI
Format-Table
Format-Wide
```

EXAMPLE

Format-Table

One object per row, one column per object property. It uses full width of display. In order to show results are they stream it guesses at the width

```
PS C:\Users\kenne\Documents\WindowsPowerShell\Example> dir | Format-Table


    Directory: C:\Users\kenne\Documents\WindowsPowerShell\Example

Mode          LastWriteTime          Length     Name
----          -------------          ------     ----
-a----        15/03/2021     16:48  12         FileOne.txt
-a----        15/03/2021     16:48  124        LongFile.txt
-a----        15/03/2021     16:47  187        MediumFile.txt
```

We can use `autosize` switch to format width better at the cost of having to wait for whole result to finish before we can display.

Format-List

One row per (object-property) pair.

```
Name           : FileOne.txt
Length         : 12
CreationTime   : 15/03/2021 16:47:25
LastWriteTime  : 15/03/2021 16:48:07
LastAccessTime : 15/03/2021 16:48:07
Mode           : -a----
LinkType       :
Target         : {}
VersionInfo    : File:
C:\Users\kenne\Documents\WindowsPowerShell\Example\FileOne.txt
                 InternalName:
                 OriginalFilename:
                 FileVersion:
                 FileDescription:
                 Product:
                 ProductVersion:
                 Debug:            False
                 Patched:          False
                 PreRelease:       False
                 PrivateBuild:     False
                 SpecialBuild:     False
                 Language:


Name           : LongFile.txt
Length         : 124
CreationTime   : 15/03/2021 16:47:31
LastWriteTime  : 15/03/2021 16:48:04
LastAccessTime : 15/03/2021 16:48:04
.
.
.
```

Format-Wide

Show single property of set of objects in concise manner.

```
dir | Format-Wide -AutoSize Length

12   124 187
```

Format-Custom

Shows object graph. Usually, we want to limit the depth.

```
PS C:\Users\kenne\Documents\WindowsPowerShell\Example> dir | Format-
Custom -Depth 1

class FileInfo
{
  LastWriteTime =
    class DateTime
    {
      Date = 15/03/2021 00:00:00
      Day = 15
      DayOfWeek = Monday
      DayOfYear = 74
      Hour = 16
      Kind = Local
      Millisecond = 143
      Minute = 48
      Month = 3
      Second = 7
      Ticks = 637514236871435962
      TimeOfDay = 16:48:07.1435962
      Year = 2021
      DateTime = 15 March 2021 16:48:07
    }
  Length = 12
  Name = FileOne.txt
}
```

Output

The following shows the supported outputs. We can use these to write to files etc.

```
Get-Command Out-* |Format-Table Name


Name
----
Out-Default
Out-File
Out-GridView
Out-Host
Out-Null
Out-Printer
Out-String
```

If I get a list of objects how to I restrict the set?

The following uses `Select-Object` to select the first in the list. As they are sorted in Descending order we get the details of the largest file

```
Get-ChildItem | Sort-Object -Property Length -Descending | Select-Object -First 1
```

How do I restrict the set of fields show on the resulting objects?

We can also use the Select-Object

```
Get-ChildItem | Sort-Object -Property Length -Descending | Select-Object -Property Name
```