第 28 章　PowerShell 备忘清单

现在是时候将遇到的一些小问题进行整理了。当你遇到什么问题时，请记住首先翻到本章进行查找。

28.1　标点符号

毫无疑问，PowerShell 命令中包含了大量的标点符号，并且大部分的标点符号在帮助文档和 PowerShell 中具有不同的含义。下面是这些标点符号在 PowerShell 中的含义。

- `（重音符）——重音符是 PowerShell 中的转义字符。它会移除紧跟在重音符后面的特定字符串的作用。例如，通常情况下，空格符是一个分隔符，这也就是在 PowerShell 中 cd C:\Program Files 会执行失败的原因。将该空格符转义，cd Program` Files，会将该空格的作用去除，仅将该符号作为文字中的一部分。这样这个命令就可以正常执行了。
- ~（波浪符）——当将~作为路径的一部分时，该字符表示当前用户的根目录，也就是在系统变量 UserProfile 中定义的值。
- ()（括号）——有两种使用场景：

-和在数学中一样，括号定义了执行的顺序。PowerShell 会优先执行括号中的命令。如果存在多重括号，则会从最里层括号向外执行。通过这种方式，可以很轻易实现：先执行一个命令，之后将该命令的输出结果传递给另外一个命令的某个参数，比如 Get-Service –ComputerName (Get-Content C:\ComputerNames.txt)。

-括号也可以被用作包含一个方法的参数。即使该方法不要求使用任何参数，也必须带有括号，比如 Change-Start-Mode('Audomatic')以及 Delete()。

- []（方括号）——在 PowerShell 中有两种使用方式：

-需要访问一个数组或者集合中某个单独的对象时，可以使用方括号来指定对应的索引号：$Services[2]表示从$Services 中获取第三个对象（请记住索引编号是从 0 开始计数的）。

-当需要将某个数据转化为特定的类型时，需要将类型包含在方括号中。例如，$My Result/3 as [INT]会将除法运算的结果转化为整数；再比如，命令[XML]$Data=Get-Content Data.XML 会读取 Data.XML 中的内容，并且尝试将该内容解析为合法的 XML 文件。

■ {}（花括号）——有三种用途：

-花括号可用作包含可执行代码或者命令块，我们称之为脚本段（Script Blocks）。该脚本段经常被作为值传递给那些可接受脚本段或者筛选块的参数：Get-Service | Where-Object{$_.Status –eq 'Running'}。

-花括号可用作包含构成哈希表的键-值对。左大括号前面总是一个“@”符号。在下面的示例中，我们使用花括号来包含哈希表的键-值对（在示例中，有两组键-值）。第二个花括号包含一段表达式的脚本段，该脚本段作为第二个键的值：$HashTable=@{l='Label';e={expression}}。

-当变量的名称中包含空格或者其他非法字符时，必须使用花括号来包含这部分信息：${My Variable}。

■ ''（单引号）——单引号可用作包含字符串（String）。PowerShell 并不会对包含在单引号中的字符串查找转义字符或者变量。

■ ""（双引号）——双引号也可用作包含字符串，但与单引号不同的是，PowerShell 会针对双引号中的字符串数据进行查找转义字符以及$字符。其中会进行针对转义字符的处理，同时$符号后面带有的字符（到下一个空格为止）会被识别为一个变量名字，并且其值会被替换掉。例如，如果变量$One 的值为“World”，同时定义$Two="Hello $One `n"，那么$Two 的值就会是“Hello World”之后再加一个回车（`n 代表一个回车键）。

■ $（美元符号）——该符号告诉 PowerShell $后面的字符（截止到下一个空格处）为一个变量的名称。但是当在使用管理变量的 Cmdlet 时，可能容易造成误解。假如$One 变量的值为 Two，然后执行 New-Variable –Name $One –Value 'Hello' 命令，会创建一个名为 Two 的变量，并且其值为“Hello”——有些人很奇怪，为什么变量的名字会是 Two。这是因为$符号告诉 PowerShell 使用$One 的值来作为新变量的名称。相对应地，New-Variable –Name One –Value 'Hello'，该命令会创建一个名为 One 的变量。

■ %（百分号）——百分号是 ForEach-Object Cmdlet 的别名，同时它也是模运算符，返回除法运算后的余数。

■ ?（问号）——问号是 Where-Object Cmdlet 的别名。

■ >（右尖括号）——该符号类似 Out-File Cmdlet 的一个别名。但是严格来讲，它并不

是一个真正的别名，但是却提供了 Cmd.exe 类型的文件重定向功能：Dir>Files.Txt。

- +-*/%（数学运算符）——这些运算符是作为标准算术运算符使用。请注意，+也可以用作字符串连接使用。
- -（破折号或者叫连字符）——可以用作连接参数名称或者其他运算符，如-ComputerName 或者-Eq。同时破折号也可以用作分离 Cmdlet 名称中的动词与名词，比如 Get-Content。另外，破折号也作为算术中的减法运算符使用。
- @（at 符号）——在 PowerShell 中有四个用途：

-可用在左花括号前面（请参阅上面的介绍花括号部分）。

-当用在括号之前时，它会包含组成数组的一串以逗号分隔的值：$Array=@(1,2,3,4)。其中的@字符与括号是可选的，因为 PowerShell 默认会将以逗号分隔的列表识别为数组。

-可以指一个 Here-String。Here-String 是指包含在单引号或者双引号中的字符串。一个 Here-String 以“@”字符作为开始和结束的标志，结束的“@”必须位于另起一行的起始位置。如果想获取更多的信息或者示例，请执行 Help About_Quoting_Rules。另外需要说明的是，Here-String 也可通过单引号进行定义。

-@也是 PowerShell 中的传递符（Splat Operator）。如果构建了一个哈希表，在哈希表中，键名称能匹配参数名称，并且键的值为参数的值，那么你就可以将该哈希表传递给一个 Cmdlet。Don 曾经为 TechNet Magazine 写过一篇关于传递（Splating）的文章（https://technet.microsoft.com/en-us/magazine/gg675931.aspx）。

- &（与符号）——这是 PowerShell 中的一个调用运算符，使得 PowerShell 可以将某些字符识别为命令，并运行这些命令。例如，$a="Dir"命令将“Dir”赋给了变量$a，然后&$a 就会执行 Dir 命令。
- ;(分号)——分号一般用作分隔 PowerShell 中同一行的两个命令：Dir;Get-Process。这个命令会先执行 Dir 命令，之后执行 Get-Process 命令。它们的执行结果会发送给一个管道，但是 Dir 命令的执行结果并不会通过管道发送给 Get-Process 命令。
- #（井号）——该符号为注释符号。跟在#之后的文字，到下一个回车之前，均会被 PowerShell 忽略掉。尖括号<>可以被用作定义一个注释块的标签，“<#”作为起始，“#>”作为结束。包含在该注释块中的所有命令均会被 PowerShell 忽略掉。
- =（等号）——等号是 PowerShell 中的赋值运算符，用来向一个变量进行赋值：$One=1。但是它不能用作数量比较，相反需要使用-Eq。另外需要记住，该运算符可以与数学运算符结合使用：$Var+=5。该命令会对$Var 变量的值增加 5。
- |（管道符）——管道符主要用作将一个 Cmdlet 的输出结果传递给另外一个 Cmdlet。第二个 Cmdlet（接收输出结果的 Cmdlet）采用管道参数绑定方法来确

定哪个参数或者哪些参数来负责接收传入的管道对象。第 9 章中对该过程进行了讲解。

- \或者/（反斜杠或斜杠）——斜杠可以作为数学表示中的除法运算符；反斜杠和斜杠也可以作为文件路径中的分隔符：C:\Windows 和 C:/Windows 路径一致。反斜杠在 WMI 筛选场景以及正则表达式中也可作为转义字符。
- .（句号）——句号有三种用途：

-句号可以被用作表示希望访问某个成员，比如一个属性或方法；再或者一个对象：$_.Status 表示访问$_占位符中对象的 Status 属性。

-它可以通过“.”引用源码来执行一段脚本，意味着该脚本运行在当前作用域下，并且该脚本定义的任何对象在脚本运行完毕之后均存在，比如.C:\myscript.ps1。

-两个“.”（..）会形成一个范围运算符，该运算符在本章后面会讲到。你也会发现，“..”也可用作表示文件系统中的当前路径的父文件夹，比如..\。

- ,（逗号）——当用在引号外面时，逗号可以用作分隔数组或者列表中的项："One",2,"Three",4。另外，它也可用作将多个静态值传递给可接收这些值的参数：Get-Process –ComputerName Server1,Server2,Server3。
- :（冒号）——冒号（严格来说应该是两个冒号）可用作访问类的静态成员。这里采用了.Net FrameWork 编程语言的概念，比如[DateTime]::Now（其实也可以使用 Get-Date 来获取相同的结果）。
- !（感叹号）——是“非”(Not)布尔运算符的别名。

我们认为，在美国键盘格式中没有被 PowerShell 使用到的应该是脱字符“^”，毕竟该符号常用于正则表达式运算。

28.2 帮助文档

帮助文档中的标点符号与 PowerShell 中相比，具有略微不同的含义。

- []——大括号，用作表达包含在大括号中的文本为可选项。比如包含在其中的所有命令（[-Name <String>]）；或者当参数是位置参数时，参数名称可选（[-Name] <String>）。也可用作表达下面两个含义：参数是可选项，并且如果指定了该参数，那么该参数可作为位置参数使用（[[-Name] <String>]）。如果你觉得有任何问题，请在命令中指定参数名称，因为这样始终是符合语法规范的。
- []——相邻的大括号表示一个参数可接受多个值（<String>[]，而非<String>）。
- <>——尖括号可用来包含数据类型，表示值的类型或者参数匹配的对象：<String>，<int>，<Process>等。

请一定要养成阅读完整帮助文档的好习惯（对 Help 命令添加-Full 参数），因为通过该命令会提供尽可能详细的信息，大多数情况下会包含示例。

28.3 运算符

PowerShell 不会使用其他编程语言使用的常规比较运算符。相反，它使用下列运算符。

- -eq——等于（-ceq 用作字符串比较，包括大小写是否一致）。
- -ne——不等于（-cne 用作字符串比较，包括大小写是否一致）。
- -ge——大于或等于（-cge 用作字符串比较，包括大小写是否一致）。
- -le——小于或等于（-cle 用作字符串比较，包括大小写是否一致）。
- -gt——大于（-cgt 用作字符串比较，包括大小写是否一致）。
- -lt——小于（-clt 用作字符串比较，包括大小写是否一致）。
- -contains——若数据集包含特定对象，则返回真（True）。（$Collection –Contains $Object。）-nocontains 表示相反含义。
- -in——若特定对象包含在数据集中，则返回真（True）。（$Object –in $Collection。）-noin 表示相反含义。

逻辑运算符可用作组合运算：

- -not——将真假值取反（!是该运算符的别名）。
- -and——如果整个表达式要为真，则所有子表达式均需要为真。
- -or——如果整个表达式要为真，则其中一个子表达式需要为真。

另外，还存在执行特定操作的运算符：

- -Join——将一个数组的元素连接为分隔的字符串。
- -Split——将一个分隔的字符串分离为一个数组。
- -Replace——将一个字符串中特定字符（串）替换为另外的字符（串）。
- -Is——若一个对象为指定类型，返回为真（True）。（$ID –Is [INT]）
- -As——将对象转化为特定类型（$ID –As [INT]）
- ..——一个范围运算符，1..10 会返回 1 到 10 的十个对象。
- -F——格式化运算符，会使用后面提供的值替换对应的占位符。（"{0},{1}" –F "Hello","World"）

28.4 自定义属性与列的语法

在多个章节中，我们曾经演示如何使用 Select-Object 来定义自定义属性，或者分别使用 Format-Table 以及 Format-List 来自定义列或列表条目。下面是对应的哈希表语法。

可以通过该语句得到每一个自定义属性或者列：

```
@{Label='Column_or_Property_Name';Expression={Value_Expression}}
```

这里的两个键“Label”和“Expression”，可以分别缩写为“l”和“e”（请注意，这里是小写的字母 l，不是数字 1）。当然，你也可以使用 n 作为键的名称。

```
@{n='Column_or_Property_Name';e={Value_Expression}}
```

在表达式中，可以使用$_占位符关联到当前对象（比如当前表中的行或者期望添加自定义属性的对象）。

```
@{n='ComputerName';e={$_.Name}}
```

Select-Object 和 Format- 的 Cmdlet 均会查找 n（或者 name 或者 label 或者 l）键和 e 键；Format- Cmdlet 也支持 Width 和 Align（仅支持 Format-Table）和 FormatString 操作。请阅读 Format-Table 命令的帮助文档，获取对应的示例。

28.5 管道参数输入

在第 9 章中我们看到，在 PowerShell 中有两种方式进行参数绑定：ByValue 和 ByPropertyName。优先使用 ByValue 方法，仅当 ByValue 方法无法执行时才会尝试使用 ByPropertyName 方法。

对 ByValue 方法而言，PowerShell 会查看放入管道中对象的类型。当然，你也可以通过 gm 命令自行查看该对象的类型名称。之后 PowerShell 会检查该 Cmdlet 中是否有参数可以接收传入的对象类型，并且检查是否有参数可以使用 ByValue 方法来接收管道输入。对一个 Cmdlet 而言，如果采用这种方式，则不可能有两个参数绑定到相同的数据类型。换句话说，你无法看到一个 Cmdlet 中有两个参数均满足如下两个条件：均可接收<String>类型的输入，均可使用 ByValue 方法实现参数绑定。

如果无法使用 ByValue 方法，那么 PowerShell 就会尝试使用 ByPropertyName 方法。在该方法中，PowerShell 仅简单查看放入管道中对象的属性，之后尝试找到某个可接收通过 ByPropertyName 方法传入对象的参数，并且要求该参数的名称与属性名称一致。例如，如果放入管道中的对象包含 Name、Status 和 ID 属性，PowerShell 会查看 Cmdlet 中是否有参数名为 Name、Status 和 ID。同时要求这些参数被标记为“可接收 ByPropertyName 管道输入”。至于如何查看是否满足条件，请阅读对应的详细帮助文档（记住，在使用 Help 命令时加上-Full 参数）。

让我们看看 PowerShell 如何实现这些功能。比如本例，假如有一个命令为 Get-Service |Stop-Service 或者是 Get-Service | Stop-Process，将其中第一个 Cmdlet 称为第一个命令，类似地，第二个 Cmdlet 称为第二个命令。PowerShell 采用下面的步骤进行工作。

（1）第一个命令产生的对象类型是什么？你可以将该 Cmdlet 输出结果通过管道传递给 Get-Member 来自行查看该信息。对那些名称由多部分字符组成的类型而言，仅需记住最后一位（比如类型名称为 System.Diagnostics.Process，仅需记住最后一位的 Process 即可）。

（2）第二个命令中是否有参数可以接收第一个命令产生的对象类型（通过查看第二个命令对应的详细帮助文档进行确定：Help <Cmdlet> -Full）？如果存在，那么再检查

该参数是否可以接收通过 ByValue 方式传入的管道对象。每个参数对应的帮助文档中的详细说明中均包含该信息。

（3）如果步骤（2）的答案是 Yes，那么第一个命令产生的完整对象就会被关联到步骤（2）中满足条件的参数。此时，所有步骤就结束了——不需要再到步骤（4）。但是如果步骤（2）的答案是“否”，那么就需要继续步骤（4）。

（4）此时需要检查第一个命令产生的对象。查看产生的对象包含什么属性。再次说明，你可以通过将第一个命令产生的对象通过管道传递给 Get-Member 来查看该信息。

（5）此时检查第二个命令的参数（此时需要重新查看详细帮助文档）。是否有参数的名称与步骤（4）中找到的属性名称一致（条件 a），并且该参数是否能接收通过 ByPropertyName 方式传入的对象（条件 b）？

（6）如果有任一参数满足步骤（5）中的 a 和 b 条件，那么第一个命令产生对象的属性就会关联到对应的第二个命令的同名参数，第二个命令就会运行。如果第一个命令产生对象的属性名称与第二个命令中可接收 ByPropertyName 方式传入对象的参数名称不一致，那么第二个命令也会运行，但是此时第二个命令并没有管道输入。

另外需要注意的是，你可以针对任意命令手动输入参数以及其值。但是此时，将会导致参数无法接收管道输入对象，即使正常情况下可以使用某种管道输入方法（不管是 ByValue 还是 ByPropertyName）。

28.6 何时使用$_

这或许是 PowerShell 中最让人费解的问题之一：什么时候才能使用$_占位符？

当 PowerShell 显式查找$_，并且准备使用其他数据填充该占位符时，可以使用$_占位符。一般来讲，这只会发生在处理管道输入的脚本段中——在这种情况下，$_占位符一次只能包含一个管道输入对象。在下面几个不同的地方会用到该占位符。

- 在 Where-Object 的筛选脚本段中：

```
Get-Service |3 Where-Object {$_.Status -eq 'Running' }
```

- 在传递给 ForEach-Object 命令的脚本段中，比如下面命令中使用的-Process 脚本段：

```
Get-WmiObject -class Win32_Service -filter "name='mssqlserver'" |
ForEach-Object -process { $_.ChangeStartMode('Automatic') }
```

- 针对过滤功能和高级功能的 Process 脚本段。我们编写的另外一本书中讨论到该部分知识——*Learn PowerShell Toolmaking in a Month of Lunches*。
- 用来创建自定义属性或者表列的哈希表表达式中，请参考 28.4 小节查看更多细节，或者阅读第 8、9、10 章中更完整的讨论。

在上面所有场景中，$_占位符仅会出现在脚本段的花括号中。那么这也是一个判断什么时候可以使用$_占位符的比较好的规则。