第 3 章　使用帮助系统

在这本书的第 1 章，我们提到由于图形用户界面具有更强的可发现性，所以更容易学习和使用。但对于像 PowerShell 这样的命令行接口-CLIs（command-line interfaces）的学习却往往要困难一些，因为它们缺乏可发现性这个特性。事实上，PowerShell 拥有出色的可发现性，但是它们并不是那么明显。其中一个主要的可发现性的功能是它的帮助系统。

3.1　帮助系统：发现命令的方法

请忍受 1 分钟的时间让我们走上讲台给你讲述下面的内容。

我们工作在一个不是特别重视阅读的行业，但是我们有一个缩写 RTFM（Read The Friendly Manual）。当我们希望他们可以“阅读易于使用的手册”时，就能巧妙地把命令传递给用户。大多数管理员更加倾向于直接上手、依赖于 GUI 工具的提示和上下文菜单等这些 GUI 的可发现性工具领会如何操作。这也是我们工作的方式。我们假设你也是以同样的方式进行工作的。但是我们来认清一点：

如果你不愿意花时间去阅读 PowerShell 的帮助文档，那么你就无法高效使用 PowerShell，也很难进一步学习如何使用它，更不用说使用它管理类似 Windows 或 Exchange 等产品，最终你无法摆脱使用 GUI 的方式。

让我们澄清一下，虽然上面一段看上去很无趣，但绝对是真理。想象一下，当你使用活动目录和计算机或是其他管理控制台时没有帮助提示、菜单、上下文菜单会怎么样。学习 PowerShell 而不去花时间去学习帮助文件也是如此。这就好像你去宜家不阅读手册就组装家具，那么你必然会经历挫折、困惑以及感到无能为力。为什么呢？

- 如果你需要执行一项任务，但是却不知道应该使用什么命令，帮助系统就可以帮助你找到这个命令，而无需使用 Google 或者 Bing。

- 如果你在运行一个命令的时候返回错误信息，帮助系统就可以告诉你如何正确运行命令，因此不再出现错误。
- 如果你想将多个命令组合在一起来执行一项复杂的任务，帮助系统就可以帮你找到哪些命令是可以和其他命令结合使用。你不需要在 Google 或者 Bing 搜索示例，只需要学习如何使用他们，以便你可以创建出自己的示例和解决方案。

我们意识到我们的讲解过于关注动手实践，但我们看到学生在课堂上或者在工作中面临的问题：如果他们能腾出几分钟坐下来、深呼吸，然后阅读帮助，90%的问题都能迎刃而解。阅读这一章，将帮助大家理解 PowerShell 的帮助文档。

从现在开始，我们鼓励你阅读帮助文档有下面几个原因。

- 虽然我们将在我们的示例中展示多个命令（我们几乎从未展示一个命令的完整功能和选项），但是你也应该阅读我们展示每个命令的帮助，这样你才会熟悉每个命令所能够完成的其他工作。
- 在本书的实验里，我们将提示你使用哪个命令完成任务，但是我们不会提示语法细节。为了完成这些实验，你必须自己使用帮助系统找到相应命令的语法。

我们向你保证，掌握帮助系统是成为 PowerShell 专家的一个关键。但你不会在帮助文档中找到每一个细节。很多高级资料并没有在帮助系统中留下文档，但为了有效的日常管理，你需要掌握帮助系统。本书会帮助你深入理解该系统以及在帮助文档中没有具体解释的部分，但只有在与帮助系统结合的情况下才会这么做。

是时候走下讲台了。

Command 对比 Cmdlet

PowerShell 包含了多种类型的可执行命令，有些叫作 Cmdlet，有些叫作函数，还有一些被称为工作流等。它们的共同点都是命令，所有这些命令都在帮助系统囊括的范围内。Cmdlet 的概念是 PowerShell 中独有的，你运行的大多数命令都属于 Cmdlet。但在谈论更通用的可执行工具时，我们会使用“命令”表示，从而保证一致性。

3.2 可更新的帮助

当你第一次使用帮助时，你也许会很惊讶，因为里面什么都没有。不要着急，我们会为你讲解。

微软在 PowerShell v3 中加入了一个新的功能，叫作“可更新的帮助”。PowerShell 可以通过互联网下载帮助文件的更新、修正和扩展。不过，为了做到可更新，微软不能把任何帮助放到安装包中。当你需要查看一个命令的帮助时，你可以得到一个自动生成的简易版的帮助，以及如何更新帮助文档的信息，如下：

```
PS C:\> help Get-Service
```

```
NAME
    Get-Service
SYNTAX
    Get-Service [[-Name] <string[]>] [-ComputerName <string[]>]
    [-DependentServices] [-RequiredServices] [-Include <string[]>]
    [-Exclude <string[]>] [<CommonParameters>]

    Get-Service -DisplayName <string[]> [-ComputerName <string[]>]
    [-DependentServices] [-RequiredServices] [-Include <string[]>]
    [-Exclude <string[]>] [<CommonParameters>]

    Get-Service [-ComputerName <string[]>] [-DependentServices]
    [-RequiredServices] [-Include <string[]>] [-Exclude <string[]>]
    [-InputObject <ServiceController[]>] [<CommonParameters>]

别名
    gsv

备注
    Get-Help 在本机无法找到关于这个 Cmdlet 命令对应的帮助文档。
    这只显示了部分帮助信息。
        -- 可使用 Update-Help 下载和安装包含这个 Cmdlet 模板的帮助文档。
        -- 可输入"Get-Help Get-Service -Online"命令或者
    输入网址 http://go.microsoft.com/fwlink/?LinkID=113332
        查看关于帮助主题的在线文档。
```

提示：一个容易被忽略的事实是通常本地并没有安装帮助，在你第一次使用帮助的时候，PowerShell 会提示你更新帮助系统。

更新 PowerShell 的帮助文档应该是你的首要任务。这些文件存储在 System32 这个目录下，这意味着你的 Shell 必须在更高特权下运行。如果在 PowerShell 的标题中没有出现“管理员”的字眼，你将会获得一个错误信息。

```
PS C:\> update-help

Update-Help : 无法更新以下模块的帮助:
'Microsoft.PowerShell.Management, Microsoft.PowerShell.Utility,
Microsoft.PowerShell.Diagnostics, Microsoft.PowerShell.Core,
Microsoft.PowerShell.Host, Microsoft.PowerShell.Security,
Microsoft.WSMan.Management' :
命令无法更新 Windows PowerShell 核心模块或 $pshome\Modules 目录中任意模块的帮助主题。若要更新
这些帮助主题，请使用“以管理员身份运行”命令启动 Windows PowerShell，然后重试运行 Update-Help。
所在位置 行:1 字符: 1
+ update-help
+ ~~~~~~~~~~~
```

```
    + CategoryInfo          : InvalidOperation: (:) [Update-Help], Except
ion
    + FullyQualifiedErrorId : UpdatableHelpSystemRequiresElevation, Micros
oft.PowerShell.Commands.UpdateHelpCommand
```

我们将之前错误信息的重点部分用粗体进行标识，该信息告诉你问题所在并如何解决。以管理员身份运行 Shell，再次运行 Update-Help 命令，几分钟内你就可以发现已经成功更新了帮助。

每隔一个月左右的时间重新获取帮助是一个很重要的习惯。PowerShell 甚至可以下载非微软发布命令的帮助文档，只要这些命令模块在合适的位置进行本地化之后加入到在线以供下载。

假如你的计算机不能连上互联网，那该怎么办呢？不要担心，首先找到一台可以上网的机器，并使用 Save-Help 命令把帮助文档下载一份到本地。然后把它放到一个文件服务器或者其他你可以访问的网络中。接着通过在 Update-Help 加上-SourcePath 参数指向刚刚下载的那份帮助文档的地址。这可以让局域网内任何计算机从中心服务器获取更新后的帮助，无需再连接互联网。

帮助文件已经开源

微软的 PowerShell 帮助文件已经在 http://github.com/powershell 开源。该网址是查看最新源码的好地方，该部分帮助可能在 PowerShell 中无法下载。

3.3 查看帮助

Windows 的 PowerShell 提供了 Get-Help 这个 Cmdlet 命令访问帮助系统。你可能看到很多示例（特别是在互联网）都是使用“Help”或“Man”（来自 UNIX，指代 Manual）关键字来代替 Get-Help。Man 和 Help 都不是原生的 Cmdlet 命令，而是对核心 Cmdlet 命令进行封装后的函数。

macOS 与 Linux 中的帮助

macOS 与 Linux 中的帮助文件，都使用操作系统传统的 Man（manual）功能进行显示，该命令会“接管”屏幕，从而显示帮助，在阅读完帮助后返回正常屏幕。

Help 的工作原理类似 Get-Help，但它可以把输出的信息通过管道传送给 More 命令。这样你就可以以分屏这样友好的方式来查看帮助的内容，而不是一次性打印出所有的帮助信息。运行 Help Get-Content 和 Get-Help Get-Content，会返回相同的结果。前者是一次一页显示，你也可以使用 Get-Help Get-Content | More 分页显示，但这需要输入更多的字符，我们通常仅使用 Help。但我们想让你知道底层实现。

注意： 从技术上来说，Help 是一个函数，而 Man 是 Help 的一个别名，或者叫昵称。但是它们返回的结果相同。我们将会在下一章讨论别名。

顺便提醒一下，有些时候你可能会讨厌分页显示，因为你想一次性获取所有的信息，但是它却一次次让你输入空格键显示余下的信息。如果你遇到这样的情况，在 Shell 控制台窗口按 Ctrl+C 组合键取消命令并立即返回到 Shell。Ctrl+C 组合键总是表示“返回”的意思，而不是“拷贝到剪切板”的意思。而在图形化 Windows PowerShell ISE 中，Ctrl+C 表示拷贝到剪切板。工具栏中有一个红色按钮“停止”，它可以用于停止正在运行的命令。

注意： 很多命令在图形化的 ISE 中不起作用，即使使用 Help 或 Man 时，它也会一次性返回所有的帮助信息，而不是一次返回一页。

帮助系统有两个主要的目标：一个是帮助你找到实现特定任务的命令，另一个就是找到命令后帮助你学会如何使用它们。

3.4 使用帮助系统查找命令

从技术上来说，帮助系统不知道 Shell 中存在哪些命令。它只知道有哪些可用的帮助主题。某些命令可能并没有帮助文档，这会导致帮助系统不能确认这个命令是否存在。幸好微软几乎发布的每个 Cmdlet 都包含一个帮助主题，这意味着你通常不会发现不同。另外，帮助系统也包含了除特定 Cmdlet 之外的其他信息，包括背景概念和其他基础信息。

与大多数命令一样，Get-Help（等同于 Help）有几个参数。其中一个最为重要的参数是-Name。该参数指定你想要访问帮助的主题名称，并且它是一个位置参数，所以你无需输入-Name，只需提供所需查找的命令名称。它也支持通配符，这让帮助系统更加容易找到命令。

例如，你想操作系统事件日志，但是你却不知道使用哪个命令，你决定搜索包含事件日志的帮助主题，可以运行下面两个命令中的一个。

```
Help *log*
Help *event*
```

第一个命令在你的计算机返回如下列表。

```
Name                                Category    Module
----                                --------    ------
Clear-EventLog                      Cmdlet      Microsoft.PowerShell.M...
Get-EventLog                        Cmdlet      Microsoft.PowerShell.M...
Limit-EventLog                      Cmdlet      Microsoft.PowerShell.M...
New-EventLog                        Cmdlet      Microsoft.PowerShell.M...
```

```
Remove-EventLog                 Cmdlet      Microsoft.PowerShell.M...
Show-EventLog                   Cmdlet      Microsoft.PowerShell.M...
Write-EventLog                  Cmdlet      Microsoft.PowerShell.M...
Get-AppxLog                     Function    Appx
Get-DtcLog                      Function    MsDtc
Reset-DtcLog                    Function    MsDtc
Set-DtcLog                      Function    MsDtc
Get-LogProperties               Function    PSDiagnostics
Set-LogProperties               Function    PSDiagnostics
about_Eventlogs                 HelpFile
about_Logical_Operators         HelpFile
```

注意：你可以注意到，前面的这个列表包含来自 Appx、MsDtc 模块的命令（和函数）等。即使你还没有将这些模块加载到内存，帮助系统也一样会显示所有模块。这可以帮助你发现电脑上被遗漏的命令。它可以发现那些安装在适当位置所有扩展中的命令。对此，我们会在第 7 章进行讨论。

前面的列表中有许多关于事件日志的函数，它们都基于“动词-名词”这个命名格式，但是最后出现了两个关于帮助主题的特殊 Cmdlets 命令却不是这种格式。这两个“about”主题提供了关于某个命令的背景信息。最后一个看起来跟事件日志没有什么关系，但是它被搜索到是因为其中有一个单词“logical”的其中一部分包含了“log”。只要有可能，我们尽量使用“*event*”或者“*log*”搜索，而不是使用“*eventlog*”，因为这样可以返回尽可能多的结果。

当发现一个 Cmdlet 有可能完成所需完成的工作时（比如说，后面示例中 Get-EventLog 看起来就是做这件事的），可以查看该 Cmdlet 的帮助文档进行确认。

```
Help Get-EventLog
```

不要忘记使用 Tab 键补全命令！它可以让你只输入部分命令名称，按下 Tab 键，接着 Shell 会完成与你刚刚输入最接近的命令。你可以连续按 Tab 键来选择其他匹配的命令。

动手实验：输入 Help Get-Ev，接着按下 Tab 键。第一次匹配到的是 Get-Event，这并不是你想要的；再次按下 Tab 键就匹配到 Get-EventLog，这就是你想要的命令。你可以敲回车键接受该命令并显示这个命令对应的帮助信息。如果你使用 ISE，你不需要敲 Tab 键。所有匹配的命令都会以列表的形式呈现，你可以选择其中一个并敲回车键，这样就完成了命令的输入。

你也可以使用最为重要的“*”通配符，它可以匹配 Help 后面零个到多个字符。如果 PowerShell 只找到一个匹配你输入的命令，它并不是以列表的形式返回这个单一项，而是直接显示这一单项的具体帮助内容。

动手实验：运行 Help Get-EventL*命令，你应该可以看到关于 Get-EventLog 的帮助信息，而不是返回一个匹配的帮助主题列表。

如果你一直跟随本书的示例进行实验，那么现在你就应该在看 Get-Eventlog 的帮助文档了。这个文档被称为概要帮助，这意味着它只有简单的命令描述和语法提示。当你需要快速了解如何使用一个命令时，这些信息非常有用，我们通过该帮助文档来进行示例讲解。

补充说明

有些时候，我们想分享的信息虽然不错，但不是至关重要的 Shell 知识。我们将把这些信息放到“补充说明”部分，正如现在这个部分。如果你跳过这段，你并没有什么损失，但是如果你进行阅读，你通常会学会以另外一种方式解决问题，或者能够更深入地了解 PowerShell。

我们前面提到过 Help 命令并不是为了搜索 Cmdlet 命令，而是为了搜索帮助主题。由于每个 Cmdlet 都有一个帮助文件，我们可以说，这些搜索到的结果集相同。但是你也可以直接使用 Get-Command 命令搜索 Cmdlet 命令（或者它的别名 Gcm）。

与 Help 这个命令一样，Get-Command 接受通配符，意味着你可以运行 Gcm *event* 查看所有名称包含“event”的命令。不管怎么样，这个返回的列表将不止包含 Cmdlet 命令，还会包含一些不一定有用的外部命令，如 netevent.dll。

一个比较好的方式是使用“-名词”或者“-动词”参数。因为只有 Cmdlet 名的名称有名词和动词，返回的结果将会限制为 Cmdlet 命令。Get-Command -noun *event*将会返回一个关于事件命令的列表；Get-Command-verb Get 将会返回一个具有检索能力的列表。你也可以使用-CommandType 参数来指定命令的类型。比如，Get-Command *log* -type Cmdlet 将会返回一个所有命令名称包含“log”的命令列表，并且这个列表不会包括任何其他扩展应用程序或者扩展命令。

3.5 帮助详解

PowerShell 的 Cmdlet 帮助文件有一些特殊的约定。从这些帮助文件中提取大量信息的关键是你需要明白自己在寻找什么，并学会更高效地使用这些 Cmdlet 命令。

3.5.1 参数集和通用参数

大部分命令可以有多种使用方式，这依赖于你需要用它们来做什么。例如，下面是 Get-EventLog 的语法帮助部分。

```
SYNTAX
   Get-EventLog [-AsString] [-ComputerName <string[]>] [-List][<CommonParameters>]

   Get-EventLog [-LogName] <string> [[-InstanceId] <Int64[]>] [-After <DateTime>]
   [-AsBaseObject] [-Before <DateTime>] [-ComputerName<string[]>] [-EntryType
    <string[]>] [-Index <Int32[]>] [-Message<string>] [-Newest <int>] [-Source
    <string[]>] [-UserName <string[]>] [<CommonParameters>]
```

注意，该命令在语法部分出现了两次，这表示这个命令提供了两个不同的参数集，你可以有两种方式使用该命令。你可能已经注意到，有些参数是这两个参数集共享的。例如，这两个参数集都包含-ComputerName 参数。但是这两个参数集总会有些差异。在这个实例中，第一个参数集提供了-AsString 和-List，这两个参数都没有出现在第二个参数集中；而第二个参数集包含许多第一个参数集中没有的参数。

下面来说明它们是如何工作的：如果你使用一个只包含在某个参数集中的参数，那么你就只能使用同一个参数集里的其他参数。如果你选择使用-List 参数，那么你能够使用的其他参数就只能是-AsString 和-ComputerName，因为存在-List 的参数集中只剩这两个参数可选。你不能添加-LogName 参数，因为它不存在于第一个参数集中。这意味着-List 和-LogName 是相互排斥的，即你不能同时使用它们，因为它们存在于不同的参数集中。

有些时候，可以只带有所有参数集中共有的参数运行命令。在这种情况下，Shell 通常会选择第一个参数集。理解你运行的命令带有的参数属于哪个参数集非常重要，因为每个参数集意味着不同的功能。

你可能已经注意到，在每个 PowerShell 的 Cmdlet 参数的结尾都有[<CommonParameters>]。不管你以何种方式使用 Cmdlet，这泛指每个 Cmdlet 命令都是使用的一组包含 8 个参数的集合。现在暂时不讨论通用参数，我们会在本书后面章节真正使用它们的时候来讨论。不过，在本章后面，如果你有兴趣，我们会告诉你哪里可以学习到更多关于通用参数的知识。

注意： 聪明的读者现在已经能够识别出我们提供示例中的变化。读者会注意到基于 PowerShell 版本的不同，Get-EventLog 的帮助布局也会不同。你甚至会看到一些新的参数，但我们所解释的基础与概念并没有变。不要因为你所看到的帮助与本书的帮助不同而卡在这里。

3.5.2 可选和必选参数

运行一个 Cmdlet 命令时，你无需提供全部参数。PowerShell 的帮助文档把可选参数放到一个方括号中。例如，[-ComputerName <string[]>]表示整个-ComputerName 参数是可选的。你可以不使用该参数，因为在没有为该参数指定一个具体值的时候，Cmdlet 会默认为本地计算机。这也就是为什么[<CommonParameters>]在方括号内，你就可以在不使用任何通用参数的情况下运行该命令。

几乎所有的 Cmdlet 命令都最少有一个可选参数。你可能永远不会需要使用其中的一些参数，以及可能日常使用其他参数。记住，当你选择一个参数时，你只需输入足够的参数名称就可以让 PowerShell 明白你所需的参数是什么。例如，-L 不能充分表示-List，因为-L 可以表示-LogName。但是-Li 是一个适合-List 的的缩写，因为其他参数名称没有以-Li 开头的。

如果运行命令时忘了指定必选参数，会发生什么呢？来看看 Get-EventLog 的帮助。例如，你可以看到-LogName 参数是必选参数，该参数并不在方括号内。尝试在没有指定日志名称的情况下运行 Get-EventLog。

动手实验：通过运行 Get-EventLog 命令而不提供任何参数。

PowerShell 会提示你需要输入必选的 LogName 参数。如果你输入类似 System 或者 Application 的参数值之后敲回车键，该命令就能正常运行。你可以按下 Ctrl+C 组合键终止该命令。

3.5.3 位置参数

PowerShell 设计者知道有些参数会被频繁地使用，而你不希望不断地输入参数名称。通常来说，参数是具有位置性的。这意味着只要你把参数值放在正确的位置，你就可以只提供这个参数值，而不需要输入具体的参数名。

有两种方式可以用于确定位置参数：通过语法概要或者通过详细的帮助文档。

在语法概要中找到位置参数

你可以在语法概要中找到第一种方式：只有参数名被方括号括起来的参数。比如，查看 Get-EventLog 的第二个参数集的前两个参数。

```
[-LogName] <string> [[-InstanceId] <Int64[]>]
```

第一个参数：-LogName。它是必选参数。我们可以识别出它是必选参数，是因为它的参数名称和参数值不在一个方括号里面。但是它的参数名称处在一个方括号内，这让它成了一个位置参数，所以我们可以只提供日志名称而不需要输入参数名称-LogName。并且由于该参数出现在帮助文档的第一个位置，所以我们知道这个日志名称是我们必须提供的第一个参数。

第二个参数：-InstanceId。它是可选的，因为它的参数名称与参数值位于同一个方括号内。在方括号内，-InstanceId 本身又处在一个方括号里，意味着它同时还是一个位置参数。它出现在第二个位置，所以我们省略这个参数名称，就必须在该位置提供一个参数值。

参数-Before（出现在语法的后面，通过运行 Help Get-EventLog 命令自行查找）是一个可选参数，因为参数名和参数值同在一个方括号里面。-Before 参数名没有单独放在方括号里，这告诉我们，当选择使用这个参数时，必须输入该参数名称（或者至少是它的别名）。

下面介绍使用位置参数时的几个技巧。

- 位置参数可以同时出现指定和不指定参数名称的情况，但是位置参数必须处在正确的位置。例如，Get-EventLog -newest 20 -Log Application 是正确的；System 会被匹配到-LogName 参数，因为这是第一个位置的参数值，20 将表示-Newest 参数值，因为你已经指定了参数名称。

- 指定参数名称总是合法的。当你这样做了，输入的顺序就变得不重要。Get-EventLog- newest 20-Log Application 是正确的，因为我们已经使用了参数名称（在这个示例中是-LogName，我们这里使用了缩写-Log）。
- 如果使用多个位置参数，不要忘了它们的位置。Get-EventLog Application 0 可以运行，Application 会附加到-LogName 参数，0 会附加到-InstanceId 参数。Get-EventLog 0 Application 会运行失败，因为 0 会附加-LogName 参数名，但是却找不到名为“0”的日志。

我们将提供一个最佳实践：总是使用参数名，直到你能顺手地使用每个 Cmdlet 并厌倦了一遍一遍输入常用的参数。在此之后，使用位置参数来节省时间。当需要把一个命令以文件的形式存储在文本文件以方便重用时，通常使用完整的 Cmdlet 名称和完整的参数名称。这样做的目的是将来可以方便地阅读和理解，因为你不需要重复输入参数名称（这毕竟也是你把命令存储在一个文件的目的），这不会增加你太多额外的输入。

在详细的帮助文档中找到位置参数

我们说通常有两种方式来位置参数。第二种方式需要你使用 Help 命令指定-full 参数来打开帮助文档。

动手实验：运行 Help Get-EventLog-full 命令。记得使用空格分页地查看帮助文档，如果你想停止查看，可以使用 Ctrl+C 组合键到达帮助文件的末尾。现在，可以通过滚动窗口重复查看整个页面。同时，如果不使用-full 参数尝试使用-ShowWindow 参数，该参数可以在客户端版本的 Windows 或带有 GUI 的 Server 版本 Windows 上执行。但是请注意成功使用-ShowWindow 的前提是底层帮助 XML 文件的质量。如果文件格式不对，你可能无法查看所有内容。注意-ShowWindow 参数无法在非 Windows 操作系统中使用。

分页查看，直到你看到类似下面关于-LogName 参数的信息。

```
-LogName <string>
    指定事件日志。输入一个事件日志的日志名称（Log 属性的值；而非 LogDisplayName）。
    不允许使用通配符。此参数是必需的。

    是否必需?                    True
    位置?                        1
    默认值
    是否接受管道输入?            False
    是否接受通配符?              False
```

在前面的例子中，你可以看到这是一个强制参数，并且是一个位置参数，同时，它出现在 Cmdlet 命令之后的第一个位置。

当学生开始使用一个 Cmdlet 命令的时候，我们总是鼓励他们把焦点放在阅读帮助上，而不只是缩写语法的提示上。阅读帮助可以让我们理解得更加详细，包括参数的使

用描述。你可以看到该参数不接受通配符，这意味着你不能提供类似 App*的参数值，你需要输入日志名称的全称，如 Application。

3.5.4 参数值

帮助文档同样给你提供了每个参数的数据类型。有些参数被称为开关参数，无需任何输入值。在缩写语法中，它们看起来如下所示。

```
[-AsString]
```

在详细语法中，它们看起来如下所示。

```
-AsString [<SwitchParameter>]
    以字符串而非对象的形式返回输出。

    是否必需?                    False
    位置?                        named
    默认值
    是否接受管道输入?            False
    是否接受通配符?              False
```

通过[<SwitchParameter>]可以确认这是一个开关参数，并不需要任何输入值。开关参数的位置可以随意放置，你必须输入参数名（或者至少是一个缩写）。开关参数总是可选的，这可以让你选择是否使用它们。

其他参数希望获得的数据类型，通常会跟在参数名称之后，并使用空格与参数名称分开（不是冒号、等号或者其他字符，虽然你时不时可能会遇到错误）。在缩写语法里面，输入的类型使用尖括号表明（我们的朋友 Jason 称之为 chi-hua-huas，在他说到这一点时带有手势比划），例如：

```
[-LogName] <string>
```

在详细语法中以相同的方式显示：

```
-Message <string>

    获取其消息中具有指定字符串的事件。可以使用此属性来搜索包含特定单词或短语的消息。允许使用通配符。
    是否必需?                    False
    位置?                        named
    默认值
    是否接受管道输入?            False
    是否接受通配符?              True
```

下面来看看通常的输入类型。

- string——一系列字母和数字，有些时候也会包含空格符。如果出现空格符，那么全部字符串必须包含在引号内。例如，类似 C:\Windows 的字符串不需要使用引号，但是 C:\Program Files 这样的字符串就需要，因为它包含了一个空格。现在，你可以交替使用单引号或者双引号，但是最好坚持使用单引号。
- Int、Int32 或 Int64——一个整数类型（整个数字不包含小数）。
- DateTime——通常，基于你本地计算机的时区配置，字符串被解释成的日期会有所不同。在美国，通过的日期格式为 10-10-2010，即月-日-年。

关于更多类型，我们将在遇到的时候再做讨论。

你也许注意到有些值包含多个方括号：

```
[-ComputerName <string[]>]
```

string 后面的方括号（我们的朋友 Json 称之为奶嘴）并不意味着某些东西是可选的。事实上，string[]意味着该参数可以接受数组、集合，或者是一个列表类型的字符串。在这种情况下，只提供一个值也符合语法。

```
Get-EventLog Security -computer Server-R2
```

但是指定多个值也符合语法。一个简单的方式是提供一个以逗号为分隔符的列表。PowerShell 把以逗号为分隔符的列表作为数组值对待。

```
Get-EventLog Security -computer Server-R2, DC4, Files02
```

再次说明，任何一个单一值中如果包含了空格，就必须使用引号。但是作为一个整体的列表，不需要使用引号，只有单一值才需要使用引号。这一点非常重要。下面的命令符合语法。

```
Get-EventLog Security -computer 'Server-R2', 'Files02'
```

如果你想为每个值都加上引号，这也是可以的（即使这些值没有一个需要引号）。但是下面将会出错：

```
Get-EventLog Security -computer 'Server-R2, Files01'
```

在该示例中，Cmdlet 命令会查找一个名称为 Server-R2，Files01 的计算机。这也许不是你想要的。

另外一种提供列表值的方式是把它们输入到一个文本文件中，每一个值一行。例如：

```
Server-R2
Files02
Files03
DC04
DC03
```

接着，你可以使用 Get-Content 这个 Cmdlet 命令来读取该文件的内容，并且发送这些内容到-computerName 参数中。你可以强制 Shell 先执行 Get-Content 命令，

这样就可以把结果送到这个参数了。

记得高中数学中()圆括号可以用来在数学表达式中指定操作的顺序。这同样适用于PowerShell。使用圆括号把命令括起来，就强制这些命令先执行。

```
Get-EventLog Application -computer (Get-Content names.txt)
```

前面一个示例展示了一个有用的技巧：我们可以把 Web 服务器、域名控制器和数据库服务器等不同类型的服务器放到一个文本文件中，接着使用这个技巧再次运行这个包含全部计算机集合的命令。

你也可以使用其他方式来输入一个列表值，包含从活动目录中读取计算机名称。这些技术会更加复杂。在学习一些 Cmdlet 命令之后，便能玩转这些技巧，我们会在后面的章节学习到。

另一种为参数（假设它是一个强制参数）指定多个值的方式是不指定参数。与所有强制参数一样，PowerShell 将提示你输入参数值。对于接受多个值的参数，你可以输入第一个值并按回车键，继续输入直到完成，最后空白处按回车键，这将告诉 PowerShell 你已经完成输入。像通常一样，如果你不想被提示输入项，可以按 Ctrl+C 组合键终止命令。

3.5.5 发现命令示例

我们倾向于通过示例学习，这就是在本书放置大量示例的原因。PowerShell 的设计者知道大部分管理员都喜欢示例，这也是他们把大量的示例放置到帮助文档的原因。如果你滚动到 Get-EventLog 帮助文档的末尾，很可能发现大量如何使用该 Cmdlet 命令的例子。

如果你只想查看示例，我们有一个简单获取到这些示例的方法：在 Help 命令中加入-example 参数，而不是使用-full 参数。

```
Help Get-EventLog -example
```

动手实验：使用这个新的参数来获取一个 Cmdlet 命令的示例。

我们喜欢这些示例，尽管其中一些会比较复杂。如果遇到一个对你来说太复杂的示例，请忽略它，并测试其他示例。或者通过一点点的尝试（总是在非生产机器上测试），看你是否知道该示例的用途以及这样用的原因。

3.6 访问“关于”主题

在本章的前面部分，我们提到 PowerShell 的帮助系统包含许多背景主题，可以用来帮助定位指定的 Cmdlet 命令。这些背景主题通常被称为“关于”主题，因为它们都是以“about_”开头的。你可能还记得在本章的前面，所有的 Cmdlet 命令都提供一个通用参数集。怎样才能更多了解这些常见的参数？

动手实验：在你继续读本书之前，确认你是否可以通过帮助系统列出公用参数。

先使用通配符。因为“common”在本书已经被多次使用，所以先从下面的关键字开始。

```
Help *common*
```

这真是一个好的关键字。事实上，这只会在帮助主题中匹配到一条记录：About_common_parameters。该主题将会自动显示，因为只有唯一一条配置的主题。浏览显示的帮助主题，你会发现如下 8 个通用参数。

```
-Verbose
-Debug
-WarningAction
-WarningVariable
-ErrorAction
-ErrorVariable
-OutVariable
-OutBuffer
```

这个帮助文档提到两个额外的“风险缓解”参数，但是并不是每个 Cmdlet 命令都提供这两个参数。

在帮助系统中，“关于”这个主题非常重要。但是，因为它们没有关联到某个特定的 Cmdlet 命令，所以很容易被人忽略。如果你运行 help about*列出所有信息，你也许会吃惊怎么有那么多额外的文档信息隐藏在 Shell 中。

3.7 访问在线帮助

PowerShell 的帮助文档是由人编写的，这意味着它们并不一定准确无误。除了更新帮助文档（你可以运行 Update-Help），微软也在其网站上发布帮助文档。PowerShell help 命令的-online 参数，使用它可以在网络中找到你所想要命令的帮助信息：

```
Help Get-EventLog -online
```

微软的 TechNet 站点承载该帮助，并且它通常比安装 PowerShell 中帮助文档要更新。如果你认为在示例或者语法中发现了一个错误，尝试查看在线版本的帮助文档。PowerShell 在线文档不会包含所有 cmdlet 的帮助信息，而是由各个产品团队负责（如 Exchange 团队、SQLServer 团队、SharePoint 团队等）共同提供帮助文档的更新。但 PowerShell 在线文档在可用的情况下，将会是内置帮助文档的补充。

我们喜欢在线帮助文档，是因为当我们在 PowerShell 输入脚本的时候，可以在另一个窗口上阅读文档（帮助文档在 Web 浏览器也能有良好的格式）。Don 通过一个简单的设置就可以使用双屏显示，效果更佳。你也可以使用我们之前提到过的-ShowWindow 这个开关参数而不是-Online 参数，在另一个窗口中打开本地帮助文档。

3.8 动手实验

注意：在本实验中，你需要在计算机中运行 PowerShell v3 或更高版本。

我们希望这一章已经帮你认识到在 PowerShell 中掌握帮助系统的重要性。现在是时候通过下面任务帮助你磨练技巧了。请记住示例答案在下一小节。使用 Help 命令寻找下面任务中的斜体字，并将它们作为提示。下面一部分步骤仅在 Windows 有效，我们会指出这些步骤。

1. 运行 `Update-Help` 并确保它执行无误。这会让你的本机下载一份帮助文档。条件是你的电脑能连上互联网，并且需要在更高特权下运行 Shell（这意味着必须在 PowerShell 的标题中出现“管理员”的字眼）。

2. 仅 Windows：哪一个 Cmdlet 命令能够把其他 Cmdlet 命令输出的内容转换到 HTML？

3. 部分仅 Windows：哪一个 Cmdlet 命令可以重定向输出到一个文件（file）或者到打印机（printer）？

4. 哪一个 Cmdlet 命令可以操作进程（processes）？（提示：记住，所有 Cmdlet 命令都包含一个名词。）

5. 你可以用哪一个 Cmdlet 命令向事件日志（log）写入（write）数据（该步骤仅在 Windows 系统有效，但你可以得到一个不同的答案）？

6. 你必须知道别名是 Cmdlet 命令的昵称。哪一个 Cmdlet 可以用于创建、修改或者导入别名（aliases）？

7. 怎么保证你在 Shell 中的输入都在一个脚本（transcript）中，怎么保存这个脚本到一个文本文件中？

8. 仅 Windows：从安全事件（event）日志检索所有的条目可能需要很长时间，你怎么只获取最近的 100 条记录呢？

9. 仅 Windows：是否有办法可以获取一个远程计算机上安装的服务（services）列表？

10. 是否有办法可以看到一个远程计算机运行了什么进程（processes）（你可以在非 Windows 操作系统找到答案，但命令本身会有区别）？

11. 尝试查看 `Out-File` 这个 Cmdlet 命令的帮助文档。通过该 Cmdlet 命令输出到文件每一行记录的默认宽度大小为多少个字符？是否有一个参数可以让你修改这个宽度？

12. 在默认情况下，`Out-File` 将覆盖任何已经存在具有相同的文件名。是否有一个参数可以预防 Cmdlet 命令覆盖现有的文件？

13. 如何查看在 PowerShell 中预先定义所有别名（aliases）列表？

14. 怎么使用别名和缩写的参数名称来写一条最短的命令，从而能检索出一台名称为 Server1 的计算机中正在运行的进程列表？

15. 有多少 Cmdlet 命令可以处理普通对象？（提示：记得使用类似“object”的单数名词好过使用类似“objects”的复数名词。）

16. 这一章简单提到了数组（arrays）。哪一个帮助主题可以告诉你关于数组的更多信息？

3.9 动手实验答案

1. Update-Help
   或者同一天执行多次：
   Update-Help -force
2. help html
   或可以尝试使用 Get-Command：
   get-command -noun html
3. get-command -noun file,printer
4. Get-command -noun process
   或：
   Help *Process
5. get-command -verb write -noun eventlog
   如果不确定名词部分是什么，使用通配符。
   help *log
6. help *alias
   或：
   get-command -noun alias
7. help transcript
8. help Get-Eventlog -parameter Newest
9. help Get-Service -parameter computername
10. help Get-Process -parameter computername
11. Help Out-File -full
    或：
    Help Out-File -parameter Width
    应该展示给你 PowerShell 默认的控制台宽度是每行 80 个字符。
12. 如果你运行 Help Out-File -full 查看参数，你将会看到-NoClobber。
13. Get-Alias
14. ps -c server1
15. get-command -noun object
16. help about_arrays
    或者可以使用通配符：
    help *array*