第 12 章　学以致用

是时候学以致用了。在本章，我们不会尝试教你任何新东西，而是使用你所学到的知识完成一个完整的示例。所以本示例必须是一个具有实际意义的示例。我们首先设定一个任务，在我们找出如何完成该任务之后，你可以跟随我们的进程。本章是本书内容的一个缩影，因为除了告诉你如何完成工作之外，我们还希望你意识到：你可以自学成才。

12.1　定义任务

首先，我们先假设你正在使用 Windows 8 或 Windows Server 2012。很明显，这两个系统已经预装了 PowerShell v3。如果你使用的 Windows 版本不是上面两个，如果可能，我们强烈推荐你下载一个试用版，或者使用例如 CloudShare.com 的服务开一个虚拟机。虽然 PowerShell v3 可以在一些老版本的 Windows 上运行，但这些版本并不像新版本那么深度管理集成。当然，使用更新版本的 Windows 或 PowerShell 也是可以的。

我们的目标是使用 PowerShell 创建一个计划任务。当我们创建完成后，可以在计划任务中看到该计划。该计划内容是每天凌晨 3 时将名称为“Accounting”的本地打印机中所有的作业删除。这么做是因为该打印机是一台老旧的设备，时不时会出现问题，导致作业无法完成。我们通过清理作业让每天有一个新的开始。

12.2　发现命令

完成任何任务的第一步都是找出哪一个命令可以完成任务。我们首先找出如何完成

打印机相关工作。这样，我们就可以通过运行命令，确保找到的命令能完成我们希望做的工作。然后，我们将该命令放入计划任务，一次只解决一个问题。

我们首先开始寻找打印机相关的命令。注意我们选择*print*而不是*printer*作为关键字。如果可能，尽量使用单词更简短的形式，从而获得更多结果。

```
PS C:\> help *print*

Name                              Category    Module
----                              --------    ------
Add-Printer                       Function    printmanagement
Add-PrinterDriver                 Function    printmanagement
Add-PrinterPort                   Function    printmanagement
Get-PrintConfiguration            Function    printmanagement
Get-Printer                       Function    printmanagement
Get-PrinterDriver                 Function    printmanagement
Get-PrinterPort                   Function    printmanagement
Get-PrinterProperty               Function    printmanagement
Get-PrintJob                      Function    printmanagement
Remove-Printer                    Function    printmanagement
Remove-PrinterDriver              Function    printmanagement
Remove-PrinterPort                Function    printmanagement
Remove-PrintJob                   Function    printmanagement
Rename-Printer                    Function    printmanagement
Restart-PrintJob                  Function    printmanagement
Resume-PrintJob                   Function    printmanagement
Set-PrintConfiguration            Function    printmanagement
Set-Printer                       Function    printmanagement
Set-PrinterProperty               Function    printmanagement
Suspend-PrintJob                  Function    printmanagement
Out-Printer                       Cmdlet      Microsoft.PowerShell.U
```

动手实验：你应该一步步跟随我们在本章运行的命令，从而确保你能看到我们所看到的结果以及信息。这里的重点不是完成任务，而是我们如何找出解决问题的方法。

上面的结果看起来有我们需要的命令，即 Get-PrintJob 和 Remove-PrintJob。如果你查看 Remove-PrintJob 的帮助，你可以看到该命令有一个接受管道输入的-InputObject 参数。这意味着我们可以获得作业对象，然后通过管道将该对象传递给 Remove-PrintJob 移除对象：

```
-InputObject <CimInstance#MSFT_PrintJob>
```

```
Required?                    true
Position?                    0
Accept pipeline input?       true (ByValue)
Parameter set name           jobObject
Aliases                      None
Dynamic?                     False
```

在生产环境的打印机上尝试“Get-PrintJob –printer "Accounting" | Remove-PrintJob”后，确认该命令可以移除所有打印作业。因此，我们完成了打印机部分的工作。下面让我们开始计划任务部分。

```
PS C:\> help *task*

Name                                Category  Module
----                                --------  ------
Disable-NetAdapterEncapsulated...   Function  NetAdapter
Enable-NetAdapterEncapsulatedP...   Function  NetAdapter
Get-NetAdapterEncapsulatedPack...   Function  NetAdapter
Set-NetAdapterEncapsulatedPack...   Function  NetAdapter
Get-CertificateNotificationTask     Cmdlet    PKI
New-CertificateNotificationTask     Cmdlet    PKI
Remove-CertificateNotification...   Cmdlet    PKI
Get-ClusteredScheduledTask          Function  ScheduledTasks
Get-ScheduledTask                   Function  ScheduledTasks
Get-ScheduledTaskInfo               Function  ScheduledTasks
New-ScheduledTask                   Function  ScheduledTasks
New-ScheduledTaskAction             Function  ScheduledTasks
New-ScheduledTaskPrincipal          Function  ScheduledTasks
New-ScheduledTaskSettingsSet        Function  ScheduledTasks
New-ScheduledTaskTrigger            Function  ScheduledTasks
```

很好——在一个名为 ScheduledTasks 的模块中发现了很多命令（当然，还有函数，但本质上是一回事）。现在让我们将搜索范围减少到仅该模块。

```
PS C:\> get-command -module scheduledtasks

Capability      Name
----------      ----
CIM             Get-ClusteredScheduledTask
CIM             Get-ScheduledTask
CIM             Get-ScheduledTaskInfo
CIM             New-ScheduledTask
```

```
CIM               New-ScheduledTaskAction
CIM               New-ScheduledTaskPrincipal
CIM               New-ScheduledTaskSettingsSet
Cmdlet, Script    New-ScheduledTaskTrigger
CIM               Register-ClusteredScheduledTask
CIM               Register-ScheduledTask
CIM               Set-ClusteredScheduledTask
CIM               Set-ScheduledTask
CIM               Start-ScheduledTask
CIM               Stop-ScheduledTask
CIM               Unregister-ClusteredScheduledTask
CIM               Unregister-ScheduledTask
```

很好，现在知道我们所需的是哪一个命令。剩下只需知道如何使用这些命令。

12.3 学习如何使用命令

我们希望创建一个新的计划任务，所以 New-ScheduledTask 看上去正是我们所需的命令。

```
PS C:\> help new-scheduledtask -full

NAME
    New-ScheduledTask

SYNTAX
    New-ScheduledTask [[-Action] <CimInstance#MSFT_TaskAction[]>]
    [[-Trigger] <CimInstance#MSFT_TaskTrigger[]>] [[-Settings]
    <CimInstance#MSFT_TaskSettings>] [[-Principal]
    <CimInstance#MSFT_TaskPrincipal>] [[-Description] <string>]
    [-CimSession <CimSession[]>] [-ThrottleLimit <int>] [-AsJob]
    [<CommonParameters>]
```

该命令不需要任何强制参数，但帮助显示-Trigger 参数用于指定任务运行的时间，-Action 参数用于指定所需完成的任务，其他参数基本上就可以忽略。帮助文档显示这两个参数接受的类型都为对象，即为 TaskTrigger 和 TaskAction 对象。所以我们需要找出如何生成这些对象。我们当然可以在帮助文档底部阅读示例代码，但这里我们小小挑战一下，因此不去阅读帮助。

通过查看 ScheduledTask 模块的任务列表，发现该列表中还包含了 New-ScheduledTaskTrigger 和 New-ScheduledTaskAction 命令，但愿可以找出我们所需要的。

```
PS C:\> help New-ScheduledTaskTrigger

NAME
    New-ScheduledTaskTrigger

SYNOPSIS

SYNTAX
    New-ScheduledTaskTrigger [-RandomDelay <TimeSpan>] [-At] <DateTime>
    -Once [<CommonParameters>]

    New-ScheduledTaskTrigger [-DaysInterval <Int32>] [-RandomDelay
    <TimeSpan>] [-At] <DateTime> -Daily [<CommonParameters>]

    New-ScheduledTaskTrigger [-WeeksInterval <Int32>] [-RandomDelay
    <TimeSpan>] [-At] <DateTime> -DaysOfWeek <DayOfWeek[]> -Weekly
    [<CommonParameters>]

    New-ScheduledTaskTrigger [-RandomDelay <TimeSpan>] [[-User] <String>]
    -AtLogOn [<CommonParameters>]

    New-ScheduledTaskTrigger [[-RandomDelay] <TimeSpan>] -AtStartup
    [<CommonParameters>]
```

当然，我们并不希望有任何随机的工作。第二个参数集包含强制的-Daily 和-At 参数，看上去正是我们所需要的。让我们试一试是否可以使用该命令创建一个触发器。

```
PS C:\> New-ScheduledTaskTrigger -daily -at 0300

WARNING: column "Enabled" does not fit into the display and was removed
Id        Frequency     Time                      DaysOfWeek
--        ---------     ----                      ----------
0         Daily         1/1/0001 12:00:00 AM
```

不，上面的结果不太对，并不是我们想要的时间，因为结果是午夜。让我们再试一次：

```
PS C:\> New-ScheduledTaskTrigger -daily -at '3:00 am'

WARNING: column "Enabled" does not fit into the display and was removed.
Id        Frequency       Time                      DaysOfWeek
--        ---------       ----                      ----------
0         Daily           4/18/2012 3:00:00 AM
```

好的，该命令可以创建我们所需的触发器。非常好。注意前面两次创建的触发器的 ID 都为 0，且该模块没有类似 Get-ScheduledTaskTrigger 的命令，这意味着 PowerShell 不会在内存中记录该对象。该命令产生一个触发器，但不会存储它。本例也说明了不要放过每一点信息（甚至是不起眼的 ID），并多加留意看到的信息。

下面开始行动：

```
PS C:\> help New-ScheduledTaskAction

NAME
    New-ScheduledTaskAction

SYNTAX
    New-ScheduledTaskAction [-Execute] <string> [[-Argument] <string>]
    [[-WorkingDirectory] <string>] [-Id <string>] [-CimSession
    <CimSession[]>] [-ThrottleLimit <int>] [-AsJob] [<CommonParameters>]
```

我们已经在 GUI 中使用过计划任务，所以我们知道-WorkingDirectory 用于设置任务运行所在的目录。-Argument 可能用于将命令行参数传递给正在运行的任务，-Execute 我们猜测用于执行希望运行的任务。让我们试一下：

```
PS C:\> New-ScheduledTaskAction -Execute 'dir'


Id               :
Arguments        :
Execute          : dir
WorkingDirectory :
PSComputerName   :
```

貌似可行。让我们把命令连起来试一下：

```
PS C:\> New-ScheduledTask -Action (New-ScheduledTaskAction -Execute 'Get-Pr
intJob -printer "Accounting"') -Trigger (New-ScheduledTaskTrigger -daily
-at '3:00 am') -Description "Reset accounting printer daily at 3am"
```

但是运行完上述命令后，我们无法在计划任务 GUI（我们终于在“metro”风格的开始屏幕中找到了计划任务）看到该任务。郁闷，我们再次回到模块的命令列表：

```
PS C:\> gcm -Module scheduledtasks
Capability      Name
----------      ----
CIM             Get-ClusteredScheduledTask
CIM             Get-ScheduledTask
CIM             Get-ScheduledTaskInfo
```

```
CIM             New-ScheduledTask
CIM             New-ScheduledTaskAction
CIM             New-ScheduledTaskPrincipal
CIM             New-ScheduledTaskSettingsSet
Cmdlet, Script  New-ScheduledTaskTrigger
CIM             Register-ClusteredScheduledTask
CIM             Register-ScheduledTask
CIM             Set-ClusteredScheduledTask
CIM             Set-ScheduledTask
CIM             Start-ScheduledTask
CIM             Stop-ScheduledTask
CIM             Unregister-ClusteredScheduledTask
CIM             Unregister-ScheduledTask
```

嗯，Register-ScheduledTask 可能是我们需要的，这看上去很有趣。以“New”开头的命令通常意味着创建某物，但我们还需要将新的任务“注册”（register），以便 Windows 能够得知该任务存在。

```
NAME
    Register-ScheduledTask

SYNTAX
    Register-ScheduledTask [-TaskName] <string> [[-TaskPath] <string>]
    [-Action] <CimInstance#MSFT_TaskAction[]> [[-Trigger]
    <CimInstance#MSFT_TaskTrigger[]>] [[-Settings]
    <CimInstance#MSFT_TaskSettings>] [[-User] <string>] [[-Password]
    <string>] [[-RunLevel] <RunLevelEnum> {Limited | Highest}]
    [[-Description] <string>] [-Force] [-CimSession <CimSession[]>]
    [-ThrottleLimit <int>] [-AsJob] [<CommonParameters>]
```

看上去和 New-ScheduledTask 非常类似，只是参数更多。如-TaskName 参数，我们推测该参数用于给任务赋予名称。我们还看到-User 和-Password，这两个参数可以在计划任务中看到。好了，让我们尝试下面命令：

```
PS C:\> Register-ScheduledTask -TaskName "ResetAccountingPrinter" -Descript
ion "Resets the Accounting print queue at 3am daily" -Action (New-Scheduled
TaskAction -Execute 'Get-PrintJob -printer "Accounting"') -Trigger (New-Sch
eduledTaskTrigger -daily -at '3:00 am')

WARNING: column "State" does not fit into the display and was removed.

TaskPath                                 TaskName
--------                                 --------
\                                        ResetAccountingPrinter
```

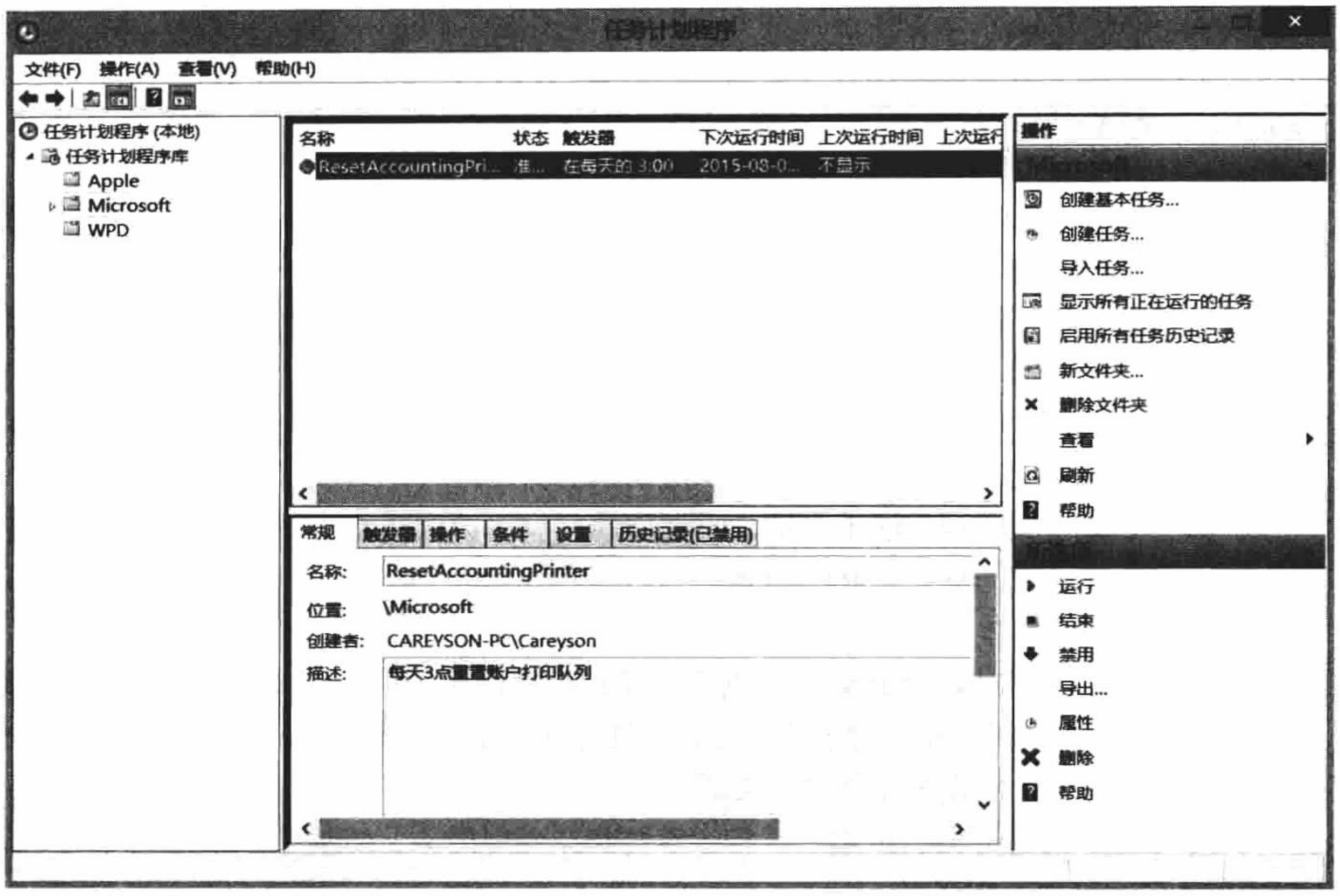

图 12.1 在计划任务的 GUI 中确认计划任务是否存在

看上去该命令完成了某些操作。回到 GUI，如图 12.1 所示，此时刚创建的任务存在了。太棒了！抱歉，当难题终于解决时，我们有点小激动。

很好，现在我们感觉就像超级英雄。但重点并不是我们完成了该项工作——而是发现如何完成工作。当然，我们也完成了这一点。当我们写本章时，我们故意选择一个别人说不可能完成且我们没有经历过的任务。因此在跟随我们学习和探索的过程中，每一点探索都可能伴随报错。

12.4 自学的一些技巧

再次说明，本书的目的是教会你如何自学——这也是本章希望阐明的。下面是一些技巧：

- 不要害怕使用帮助并确保阅读示例。我们不止一次强调过这一点，但好像没人愿意听。我们仍然会看到很多学生在我们眼皮底下使用 Google 寻找示例。为什么那么害怕帮助文档？如果你都愿意读别人的博客了，为什么不先尝试在帮助文档中阅读示例？

- 请注意，在屏幕上，每一点信息可能都非常重要——请不要跳过不是你目前正在寻找的信息。你很容易这样做，但请不要这么做。要查看每一部分信息，并尝试发现该信息的用处，以及能使用该信息能够推算出什么。当每次创建触发器的 ID 都为 0，而不是每次创建触发器都有一个连续递增的触发器时，我们可以安全地确认 PowerShell 不会将该触发器存到某个列表。这还意味着我们需要将触发器传递给某个父命令，而不是先创建它供后续使用。
- 不要害怕失败，希望你弄一个虚机，然后在虚拟里玩 PowerShell。学生们经常会问类似这类问题："如果我做了这个和那个，会发生什么？"我们的回答往往是"不知道，自己试试"。在虚拟机做实验是一个好办法，最坏的情况也只不过是将虚机回滚到某个快照点，对吧？所以无论做什么，都请试一试。
- 如果尝试一种方法不奏效，不要挠墙——请尝试其他方法。本例中我们指定了错误的时间格式 0300，且懒得去读示例中的正确写法。我们选择了将'3:00 am'作为字符串，而不是不停尝试 03:00、03:00:00 等格式。
- 我们利用对 GUI 计划任务的直觉猜出一些命令，比如-Execute 开关。请不要让使用命令行 Shell 导致你忘记过去使用 Windows 的经验——请尝试将你所做的和你过去所做的工作产生关联。

很明显，随着时间的流逝，所有的事情都会变得简单。请耐心并保持练习——但同时在学习过程中不忘思考。

12.5 动手实验

注意：对于本次动手实验来说，你需要运行 PowerShell v3 或更新版本 PowerShell 的计算机。

下面该轮到你了。我们假设你正在使用虚拟机或其他你可以假借学习的名义搞乱的环境。请不要在生产环境和运行关键系统的计算机上进行实验。

Windows 8 和 Windows Server 2012 包含一个使用文件共享的模块。你的任务是创建一个名称为"LABS"的目录，并共享该目录。为了练习的简便，先假设该目录和共享不存在。先不用管 NTFS 的权限问题，但请确保共享目录的权限设置为所有人拥有读/写权限，并且本地管理员拥有完全控制权。由于共享的主要是文件，你还希望为文档设置共享缓存。你的脚本还应该展示新建的共享及其权限。