# 附录　复习实验

当你完成这本书中指定的章节和实验后，可以继续完成本篇附录中提供的复习实验。对于你的学习过程来说，复习是一种很好的休息方式，同时可以巩固你已经学到的最为重要的要点。参考答案在本附录的末尾部分。

因为这些实验任务中的一部分实验说明命令较为复杂，所以我们已经将这些复杂的说明命令分解为独立的任务小节。同时为了帮助你完成实验，在每个实验开端，我们也提供了一个提示清单来提示你，包括你可能会需要的特定命令、帮助文件和语法。

## 复习实验 1：第 1~6 章

**注意：**为了完成这些实验，你需要一台运行 PowerShell v3 或更新版本的 PowerShell 的计算机。在打算完成这些实验之前，你应该先完成这本书中的第 1~6 章的实验。

**提示：**

- `Sort-Object`
- `Select-Object`
- `Import-Module`
- `Export-CSV`
- `Help`
- `Get-ChildItem (Dir)`

### 任务 1

运行一个命令，从而显示应用程序事件日志中最新的 100 个条目，不要使用 `Get-WinEvent`。

### 任务 2

写一个仅显示前五个最消耗虚拟内存（VM）进程的命令。

### 任务 3

创建一个包含所有服务的 CSV 文件，只需要列出服务名称和状态。所有处于运行状态的服务位于停止状态的服务之前。

### 任务 4

写一个命令行，将 BITS 服务的启动类型变更为手动。

### 任务 5

显示你计算机中所有文件名称为 `Win*.*`的文件，以 `C:\`开始。注意：为了完成该实验，你可能需要去实验和使用一个 Cmdlet 命令的新参数。

### 任务 6

获取一个 C:\Program Files 的目录列表。包含所有的子文件夹，把这些目录列表放到位于 C:\Dir.txt 的文本文件内（记住使用“>”，或者 `Out-File` Cmdlet）。

### 任务 7

获取最近 20 条安全事件日志的列表，将这些信息转化成 XML 格式。不要在硬盘上创建文件，而是把 XML 在控制台窗口直接显示出来。

**注意：**该 XML 可以作为一个单独的原生对象显示，而不是以一个原始的 XML 数据。这没问题。那也是 PowerShell 展示 XML 的方式。如果你喜欢，你可以将 XML 对象通过管道传递给 `Format-Custom` 命令，从而查看 XML 展开为对象层级的形式。

### 任务 8

获取一个服务列表，并将其导出到以 C:\services.csv 命名的 CSV 文件内。

### 任务 9

获取一个服务列表，仅保留服务名称、显示名称和状态，然后将这些信息发送到一个 HTML 文件中。在 HTML 文件中的服务信息表格之前显示“Installed Services”。如果可以，将安装服务显示在<H1>这个 html 标签中。在 Web 浏览器中验证该文件是否正确。

### 任务 10

为 Get-ChildItem 创建一个新的别名 D。仅将别名导出到一个文件里。关闭这个 Shell，然后打开一个新的控制台窗口。把别名导入到新的 Shell 中。确认能够通过运行 D 并且获得一个目录列表。

### 任务 11

显示类别为“Hotfix”或“Update”的补丁，结果中不包含安全更新。

### 任务 12

运行一个用于展示 Shell 所在的当前目录的命令。

### 任务 13

运行一个命令，展示最近你在 Shell 中运行过的命令。从中查找你在任务 11 中所运行的命令。将这两个命令通过管道传输符进行连接，重新运行任务 11 的命令。

换句话说，假如 Get-Something 是一个获取历史命令的命令，5 是任务 11 的命令 ID 号，并且 Do-Something 是运行历史命令的命令，运行如下。

```
Get-Something          -id 5 | Do-Something
```

当然，上面的命令并不是正确的命令，你需要找到正确的命令。

**提示：**你所需寻找的两个命令名词部分相同。

### 任务 14

运行命令修改安全事件日志，使得在需要时可以通过覆盖旧日志的方式新增日志。

### 任务 15

通过使用 New-Item Cmdlet 来创建一个名称为 C:\Review 的新目录。这与运行 Mkdir 并不同；New-Item 命令需要知道你所想要创建的新项目是什么类型。请阅读该命令的帮助信息。

### 任务 16

显示该注册码的内容。

```
HKCU:\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders
```

**注意：**“User Shell Folders”与真正意义上的目录并不一样。如果你改变该“目录”，你将不能在目录清单中看到任何条目。User Shell Folders 是一个项目，其包含的是项目属性。有一个 Cmdlet 能展示属性项（尽管命令的名词部分是单数形式，而不是复数形式）。

### 任务 17

找出（但是请不要运行）能完成如下功能的命令。

- 重启电脑。
- 关闭电脑。
- 从一个工作组或者域内移除一个电脑。
- 恢复一个电脑系统，并重建检查点。

### 任务 18

你认为什么命令可以改变一个注册表值？提示：该命令与任务 16 中的命令有相同的名词部分。

## 复习实验 2：第 1 ~ 14 章

**注意：**为了完成这些实验，你需要一台运行 PowerShell v3 或更新版本的 PowerShell 的计算机。在打算完成这些实验之前，你应该先完成这本书中的第 1 ~ 14 章的实验。

**提示：**

- `Format-Table`
- `Invoke-Command`
- `Get-Content(or Type)`
- `Parenthetical commands`
- `@{label='column_header';expression={$_.property}}`
- `Get-WmiObject`
- `Where-Object`
- `-eq -ne -like -notlike`

### 任务 1

在一个表格中展示一个正在运行的进程的列表，其中只包含进程名称与 ID 号。不要让该表格在两列之间有较大的空白区域。

### 任务 2

运行如下命令。

```
Get-WmiObject -class Win32_systemdriver
```

现在再次执行该命令，但将输出结果格式化为一个列表，该列表包含驱动短名称、驱动的显示名称、驱动文件路径、启动模式以及当前状态。将路径属性的列标题显示为 Path，而不是其原本的名称。

### 任务 3

让两台电脑（也可以使用 Localhost 两次）运行如下命令。

```
Get-PSProvider
```

使用远程处理去做，确保输出结果包含计算机名称。

### 任务 4

使用 Notepad 创建一个名称为 C:\Computers.txt 的文件。在文件中写入如下内容。

```
Localhost
Localhost
```

你应该确保上述两个名称各自独占一行——总共 2 行。保存文件并关闭记事本。然后写一个命令列出正在电脑上运行的服务名称，并将其写入到 C:\Computer .txt 文件中。

### 任务 5

查询 `Win32_LogicalDisk` 的所有实例。仅显示 DriveType 属性中包含 3 且有百分之五十以上的可用磁盘空间的实例。你可能需要调整可用空间百分比参数从而在你的计算机上能够得到输出结果。

**提示：** 计算可用空间百分比，公式为 `freespace/size * 100`。

注意，`Get-WmiObjectcannot` 的过滤参数中无法包含数学表达式。

### 任务 6

显示在 root\CIMv2 的命名空间下的所有的 WMI 类列表，仅显示以“win32”开头的 WMI 类名称。

### 任务 7

在列表中显示所有 StartMode 是 Auto 且 State 属性不是 Running 的 Win32_Service 实例。

### 任务 8

找到一个能发送 E-mail 信息的命令。这个命令的必要参数都是什么？

### 任务 9

运行一个显示 C:\下目录权限的命令。你会发现以列表的形式显示输出结果会更易于阅读。

### 任务 10

运行一个可以显示所有 C:\Users 下子文件夹权限的目录，仅包含直接子文件夹，不需要去递归所有的文件和文件夹。你只需要把一个命令的结果通过管道传输给另一个命令即可实现。然后重复该过程从而显示隐藏文件夹的权限目录。

### 任务 11

找到一个可以使用其他凭据而不是当前登录用户的凭据启动记事本的命令。

### 任务 12

运行一个命令，使 Shell 暂停或者闲置 10 秒。

### 任务 13

你能找到解释 Shell 的各种运算符的帮助文件吗？

### 任务 14

使用 `Get-Winevent`，显示所有拥有条目的日志文件列表，并根据所包含日志文件的多少，按照降序排序输出结果。

### 任务 15

运行如下命令。

```
Get-CimInstance -Classname Win32_Processor
```

了解该命令的默认输出结果。现在修改这个命令，使得输出结果在表格中显示。表格

内容应该包含每个处理器的核心数、制造商和名称，也包括一个列名为“MaxSpeed”的列，该列表示处理器的最大时钟频率。

### 任务 16

运行如下命令。

```
Get-CimInstance -Classname Win32_Process
```

了解该命令的默认输出结果。然后将该输出结果通过管道传递给 `Get-Member` 命令。现在，将该命令修改为仅显示在峰值情况下工作集超过“100000”的进程，仅显示进程名称、路径以及所有峰值属性。

### 任务 17

如果你正在使用 PowerShell 5 或更新版本，使用 Find-Module 命令发现带有 Network 标签的包。显示模块的名称、版本以及描述。

## 复习实验 3：第 1 ~ 19 章

**注意：**为了完成这些实验，你需要一台运行 PowerShell v3 或更新版本的 PowerShell 的计算机。在打算完成这些实验之前，你应该先完成这本书中的第 1 ~ 19 章的实验。

从回答下列问题开始。

1. 你会使用哪一个命令启动一个完全在你本地计算机运行的作业？
2. 你会使用哪一个命令启动一个作业的内容被远程计算机处理但由本地计算机调整的作业？
3. `${computer name}`是一个合法的变量名称吗？
4. 你会如何展示由当前 Shell 定义的变量列表？
5. 哪一个命令可以被用来提示用户输入？
6. 哪一个命令可以被通常用于生成显示在屏幕上的输出结果，但也可以被重新转为多种其他输出格式？

现在完成以下任务。

### 任务 1

创建一个处于运行状态的进程列表，该列表应该仅包含进程名称、ID、VM 和 PM。VM 与 PM 的值显示单位为 MB。把这个列表放入一个名称为 C:\Procs.html 的 HTML 文件中。确保 HTML 文件有一个标题为“Current Processes”。在浏览器中显示文件，并把标题显示在浏览器窗口的标题栏中。为 VM 属性计算 MB 并以整数显示结果的公式是类似`$_.VM / 1MB as [int]`。然后尝试将该 HTML 文件重新导入回 PowerShell。

### 任务 2

使用 WMI 或 CIM 命令创建一个包含所有你的电脑上的服务的制表符分隔文件，命名为 C:\Services.tdf。"`t"（在双引号之间的反撇号 t）是 PowerShell 为水平制表符使用的转义字符。文件中仅包含服务的名称、显示名称和状态。

### 任务 3

首先，提示用户输入一个计算机名称并将结果存入一个变量。然后使用 CIM 命令查询一个计算机（使用变量）的操作系统名称、版本号、上次启动时间以及运行时间。在结果中包含计算机名称。你可以通过当前时间与上次启动时间计算出计算机的运行时间。

### 任务 4

将任务 3 的命令转换为参数化脚本。显而易见，computername 是一个不错的候选参数。包含一个名称为 CN 的别名，并将其设置为必要参数。输出结果应该显示和任务 3 相同的属性，但你或许希望操作系统名称的显示名称更加优雅。

### 任务 5

使用 WMI 的 Win32_Product 类找出所有已安装的产品。该命令可能会花费较长时间，因此请将其设置为一个后台作业。当该命令完成后，获取结果，并在 gridview 中显示产品名称、公司、版本号、安装日期以及安装区域。

## 答案

## 复习实验 1

### 任务 1

```
Get-EventLog -LogName Security -Newest 100
```

### 任务 2

```
Get-Process | Sort -Property VM -Descending | Select -First 5
```

### 任务 3

```
Get-Service | Select -Property Name,Status |
     Sort -Property Status -Descending |
Export-CSV services.csv
```

### 任务 4

```
Set-Service -Name BITS -StartupType Automatic
```

### 任务 5

```
Get-ChildItem -Path C:\ -Recurse -file -Filter 'Win*.*'
```

### 任务 6

```
Get-ChildItem -Path 'c:\program files' -recurse | Out-File c:\dir.txt
```

### 任务 7

```
Get-EventLog -LogName Security -Newest 20 | ConvertTo-XML
```

### 任务 8

```
Get-EventLog -list | Select Log,MaximumKilobytes,OverflowAction |
     convertto-csv
```

### 任务 9

```
Get-Service | Select -Property Name,DisplayName,Status |
ConvertTo-HTML -PreContent "<H1>Installed Services</H1>" -title
     "Service Report" | Out-File c:\services.html
```

### 任务 10

```
New-Alias -Name D -Value Get-ChildItem -PassThru | Export-Alias c:\alias.xml
```

在打开一个新的 PowerShell 窗口后：

```
Import-Alias c:\alias.xml
D
```

### 任务 11

```
get-hotfix -description "Update","Hotfix"
```

### 任务 12

```
Get-Location or its alias pwd.
```

### 任务 13

```
Get-History
```

在执行完该命令后，找到你为完成任务 11 所执行的命令。你需要该命令的 ID 号，你需要将 id 号替换下面命令的 x。

```
Get-History -id x | Invoke-History
```

### 任务 14

```
Limit-EventLog -LogName Security -OverwriteAction OverwriteAsNeeded
```

### 任务 15

```
New-Item -Name C:\Review -Type Directory
```

### 任务 16

```
Get-ItemProperty -Path
    'HKCU:\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell
    Folders'
```

### 任务 17

- Restart-Computer
- Stop-Computer
- Remove-Computer
- Restore-Computer

### 任务 18

```
Set-ItemProperty
```

## 复习实验 2

### 任务 1

```
Get-Process | Format-Table -Property Name,ID -AutoSize
```

### 任务 2

```
Get-wmiobject -class win32_systemdriver | select -property Name,Displayname,
    @{Name="Path";Expression={$_.pathname}},StartMode,State
```

### 任务 3

```
Invoke-Command -ScriptBlock { Get-PSProvider } -computerName
    Computer1,Computer2
```

### 任务 4

```
Get-Service -computerName (Get-Content C:\Computers.txt)
```

### 任务 5

```
Get-WmiObject -class Win32_LogicalDisk -Filter "drivetype=3" |
     Select DeviceID,Size,Freespace,
     @{Name="PctFree";Expression = {($_.freespace/$_.size)*100}} |
     where {$_.PctFree -gt 50}
```

### 任务 6

```
get-cimclass -classname win32*
```

### 任务 7

```
Get-WmiObject -class Win32_Service -filter "StartMode='Auto' AND
     State<>'Running'"
```

该表达式也同样可以生效，但并不是推荐的最佳实践，因为该表达式不是左过滤。

```
Get-WmiObject -class Win32_Service |
     Where-Object { $_.StartMode -eq 'Auto' -and $_.State -ne 'Running' }
```

### 任务 8

Send-MailMessage（读取完整帮助，找出必要参数）

### 任务 9

```
Get-ACL -Path C:\ | format-list
```

### 任务 10

```
Get-ChildItem -path C:\Users | Get-ACL
Get-ChildItem -path C:\Users -Directory -Hidden | Get-ACL
```

### 任务 11

```
Start-Process
```

### 任务 12

```
Start-Sleep -seconds 10
```

### 任务 13

```
Help *operators*
```

### 任务 14

```
get-winevent -ListLog * | where {$_.recordcount -gt 0} | sort RecordCount -
    Descending
```

### 任务 15

```
Get-CimInstance -classname Win32_Processor |
    Select-Object -property Manufacturer,NumberOfCores,Name,@{
    name='MaxSpeed';expression={$_.MaxClockSpeed}}
```

### 任务 16

```
Get-WmiObject -class Win32_Process -filter "PeakWorkingSetSize >= 100000" |
    Select Name,ExecutablePath,Peak*
```

### 任务 17

```
find-module -tag network | Sort Name | Select Name,Version,Description
```

## 复习实验 3

1. Start-Job
2. Invoke-Command
3. Yes
4. Read-Host
5. Write-Output

### 任务 1

```
Get-Process | Select-Object -property Name,ID,VM,PM |
    ConvertTo-HTML -Title "Current Processes" | Out-File C:\Procs.html
```

### 任务 2

```
Get-CimInstance -classname win32_service |
    Select Name,State,StartMode,Startname |
    Export-CSV c:\services.tdf -Delimiter "`t"
```

### 然后尝试

```
import-csv C:\services.tdf -Delimiter "`t"
```

### 任务 3

```
$computer = Read-Host "Enter a computername"
```

```
Get-CimInstance -ClassName Win32_Operatingsystem -CimSession $computer |
     Select Caption,Version,LastBootUptime,
@{Name="Uptime";Expression={(Get-Date) - $_.lastBootUpTime}},
PSComputername
```

### 任务 4

这里给出一个可能的脚本版本：

```
[cmdletbinding()]
Param(
[Parameter(Mandatory=$True,HelpMessage = "Enter a computer name")]
[Alias("CN")]
[string]$Computername
)

Write-Verbose "Getting Operating system information from $Computername."
Get-CimInstance -ClassName -CimSession $computername |
Select @{Name="OS";Expression={$_.Caption}},Version,LastBootUptime,
@{Name="Uptime";Expression={(Get-Date) - $_.lastBootUpTime}},
@{Name="Computername";Expression={$_.PSComputername}}
Write-Verbose "Done."
```

### 任务 5

首先创建一个作业，这是一种方法。

```
get-wmiobject win32_product -asjob
```

然后接收结果。

```
$prod = Receive-job 31 -Keep
```

最后，处理这些结果。

```
$prod | Select Name,Vendor,InstallDate,InstallLocation |
     Out-Gridview -Title "My Products"
```