Chapter 3 Lab Answers

1. First, run Update-Help and ensure it completes without errors. That will get a copy of the help on your local computer. This requires an Internet connection, and requires that the shell be running under elevated privileges (which means it must say "Administrator" in the shell's window title bar).

```
Update-Help
or if you run it more than once in a single day:
Update-Help –force
```

2. Can you find any cmdlets capable of converting other cmdlets' output into *HTML*?

```
Help html
Or you could try with Get-Command
get-command -noun html
```

3. Are there any cmdlets that can redirect output into a *file*, or to a *printer*?

```
get-command -noun file,printer
```

4. How many cmdlets are available for working with *processes*? (Hint: remember that cmdlets all use a singular noun.)

```
Get-command –noun process
Or
Help *Process
```

5. What cmdlet might you use to *write* to an event *log*?

```
get-command -verb write -noun eventlog
or if you weren't sure about the noun, use a wildcard
help *log
```

6. You've learned that aliases are nicknames for cmdlets; what cmdlets are available to create, modify, export, or import *aliases*?

```
Help *alias
Or
get-command -noun alias
```

7. Is there a way to keep a *transcript* of everything you type in the shell, and save that transcript to a text file?

```
Help transcript
```

8. It can take a long time to retrieve all of the entries from the Security *event* log. How can you get just the 100 most recent entries?

help Get-EventLog -Parameter Newest

9. Is there a way to retrieve a list of the *services* that are installed on a remote computer?

help Get-Service -Parameter computername

10. Is there a way to see what *processes* are running on a remote computer?

Help Get-Process –Parameter computername

11. Examine the help file for the `Out-File` cmdlet. The files created by this cmdlet default to a width of how many characters? Is there a parameter that would enable you to change that width?

Help Out-File –full

Or

Help Out-File –Parameter Width

Should show you 80 characters as the default for the PowerShell console. You would use this parameter to change it as well.

12. By default, `Out-File` will overwrite any existing file that has the same filename as what you specify. Is there a parameter that would prevent the cmdlet from overwriting an existing file?

If you run: Help Out-File –full and look at parameters you should see –NoClobber.

13. How could you see a list of all *aliases* defined in PowerShell?

Get-alias

14. Using both an alias and abbreviated parameter names, what is the shortest command line you could type to retrieve a list of running processes from a computer named Server1?

ps –c server1

15. How many cmdlets are available that can deal with generic objects? (Hint: remember to use a singular noun like "object" rather than a plural one like "objects").

get-command -noun object

16. This chapter briefly mentioned *arrays*. What help topic could tell you more about them?

help about_arrays

or if you weren't sure, use wildcards

help *array*

Chapter 4 Lab

Using just what you learned in this chapter, and in the previous chapter on using the help system, complete the following tasks in Windows PowerShell:

1. Display a list of running processes.

   Get-Process

2. Display the 100 most recent entries from the Application event log (don't use `Get-WinEvent` for this – We've shown you another command that will do this task).
   Get-EventLog -Newest 100 -LogName "Application"

3. Display a list of all commands that are of the "cmdlet" type (this is tricky – we've shown you `Get-Command`, but you're going to have to read the help to find out how to narrow down the list as we've asked).

   Get-Command -Type Cmdlet

4. Display a list of all aliases.

   Get-alias

5. Make a new alias, so that you can run `d` to get a directory listing.
   Set-Alias -Name d -Value dir

6. Display a list of services that begin with the letter "M." Again, read the help for the necessary command – and don't forget that "*" is a near-universal wildcard in PowerShell.

   Get-Service "M*"

7. Display a list of all Windows Firewall rules. You'll need to use `Help` or `Get-Command` to discover the necessary cmdlet!
   Get-NetFirewallRule

8. Display a list only of inbound Windows Firewall rules. Same cmdlet as above, but you'll need to read its help to discover the necessary parameter and its allowable values

   Get-NetFirewallRule -Direction Inbound

Chapter 5 Lab

Complete the following tasks:

1. In the registry, go to HKEY_CURRENT_USER\software\microsoft\Windows\currentversion\explorer. Locate the Advanced key, and set its DontPrettyPath property to 1.

   ```
   cd HKCU:\software\microsoft\Windows\currentversion\explorer
   cd advanced
   Set-ItemProperty -Path . -Name DontPrettyPath -Value 1
   ```

2. Create a zero-length file named C:\Test.txt (use `New-Item`).

   ```
   New-Item -Name test.txt -ItemType file
   ```

3. Is it possible to use `Set-Item` to change the contents of C:\Test.txt to TESTING? Or do you get an error? If you get an error, why?

   The file system provider does not support this action.

4. What are the differences between the `-Filter`, `-Include`, and `-Exclude` parameters of `Get-ChildItem`?

   Include and exclude must be used with –Recurse or if querying a container. Filter uses the PSProviders filter capability which not all Providers support. For example, you could use DIR –filter in the file system but not in the registry. Although you could use DIR –include in the registry to achieve almost the same type of filtering result.

Chapter 6 Lab

1. Create two similar, but different, text files. Try comparing them using `Diff`. To do so, run something like this: `Diff -reference (Get-Content File1.txt) -difference (Get-Content File2.txt)`. If the files have only one line of text that's different, the command should work.

   ```
   PS C:\> "I am the walrus" | out-file file1.txt
   PS C:\> "I'm a believer" | out-file file2.txt
   PS C:\> $f1=get-content .\file1.txt
   PS C:\> $f2=Get-Content .\file2.txt
   [quark]: PS C:\> diff $f1 $f2

   InputObject                  SideIndicator
   -----------                  -------------
   I'm a believer                 =>
   I am the walrus                 <=
   ```

2. What happens if you run `Get-Service | Export-CSV services.csv | Out-File` from the console? Why does that happen?

   If you don't specify a file name with Out-File you'll get an error. But even if you do Out-File won't really do anything because the file is actually created by Export-CSV.

3. Apart from getting one or more services and piping them to `Stop-Service`, what other means does `Stop-Service` provide for you to specify the service or services you want to stop? Is it possible to stop a service without using `Get-Service` at all?

   Stop-Service can accept one or more service names as a parameter values for the –Name parameter. For example, you could run:
   Stop-Service spooler

4. What if you wanted to create a pipe-delimited file instead of a comma-separated file? You would still use the `Export-CSV` command, but what parameters would you specify?

   ```
   get-service  | Export-Csv services.csv -Delimiter "|"
   ```

5. Is there a way to eliminate the `#` comment line from the top of an exported CSV file? That line normally contains type information, but what if you wanted to omit that from a particular file?

6. `Export-CliXML` and `Export-CSV` both modify the system, because they can create and overwrite files. What parameter would prevent them from overwriting an existing file? What parameter would ask you if you were sure before proceeding to write the output file?

   ```
   get-service  | Export-Csv services.csv –noclobber
   get-service  | Export-Csv services.csv -confirm
   ```

7. Windows maintains several regional settings, which include a default list separator. On U.S. systems, that separator is a comma. How can you tell `Export-CSV` to use the system's default separator, rather than a comma?

   ```
   get-service  | Export-Csv services.csv -UseCulture
   ```

Chapter 7 Lab

As always, we're assuming that you have the latest version of Windows (client or server) on a computer or virtual machine to test with.

For this lab, you only have one task: run the Networking troubleshooting pack. When you successfully do so, you'll be asked for an "Instance ID;" just hit Enter. Then, run a Web Connectivity check, and ask for help connecting to a specific Web page. Use http://videotraining.interfacett.com as your test URL. Hopefully, you'll get a "No problems were detected" report, meaning you ran the check successfully.

To accomplish this task, you'll need to discover a command capable of getting a troubleshooting pack, and one capable of executing a troubleshooting pack. You'll also need to discover where the packs are located and how they're named. Everything you need to know is in PowerShell, and the help system will find it for you.

That's all the help you get!

Here is one way to approach this:

get-module *trouble* -list

import-module TroubleShootingPack

get-command -Module TroubleShootingPack

help get-troubleshootingpack –full

help Invoke-TroubleshootingPack -full

dir C:\windows\diagnostics\system

$pack=get-troubleshootingpack C:\windows\diagnostics\system\Networking

Invoke-TroubleshootingPack $pack

Enter

1

2

http://videotraining.interfacett.com

Chapter 8 Lab

This chapter has probably covered more, and more difficult, new concepts than any chapter so far. Hopefully we were able to make it all make sense, but these exercises should help you cement everything. See if you can complete them all, and remember that there are companion videos and sample solutions at MoreLunches.com. Some of these tasks will draw on skills you learned in previous chapters, as a way of refreshing your memory and keeping you sharp.

1. Identify a cmdlet that will produce a random number.

   Get-Random

2. Identify a cmdlet that will display the current date and time.

   Get-Date

3. What type of object does the cmdlet from task #2 produce? (What is the *type name* of the object produced by the cmdlet?)

   System.DateTime

4. Using the cmdlet from task #2 and `Select-Object`, display only the current day of the week in a table like this (caution: The output will right-align, so make sure your PowerShell window doesn't have a horizontal scroll bar):

   Get-Date | select DayofWeek

5. Identify a cmdlet that will display information about installed hotfixes.

   Get-Hotfix

6. Using the cmdlet from task #5, display a list of installed hotfixes. Sort the list by the installation date, and display only the installation date, the user who installed the hotfix, and the hotfix ID.

   Get-HotFix | Sort InstalledOn | Select InstalledOn,InstalledBy,HotFixID

7. Repeat task #6, but this time sort the results by the hotfix description, and include the description, the hotfix ID, and the installation date. Put the results into an HTML file.

   Get-HotFix | Sort Description | Select Description,InstalledOn,InstalledBy,HotFixID | ConvertTo-Html -Title "HotFix Report" | Out-File HotFixReport.htm

8. Display a list of the 50 newest entries from the Security event log (you can use a different log, such as System or Application, if your Security log is empty). Sort the list so that the oldest entries appear first, and so that entries made at the same time are sorted by their index. Display the index, time, and source for each entry. Put this information into a text file (not an HTML file, just a plain text file). You may be tempted to use `Select-Object` and its `-first` or `-last` parameters to achieve this; don't. There's a better way. Also, avoid using `Get-WinEvent` for now – there's a better cmdlet to work with for this particular task.

   Get-EventLog -LogName System -Newest 50 | Sort TimeGenerated,Index | Select Index,TimeGenerated,Source | Out-File elogs.txt

Chapter 11 Lab

Remember that Where-Object isn't the only way to filter, and it isn't even the one you should turn to first. We've kept this chapter a bit shorter so that you can have more time to work on hands-on examples, so following the principle of filter left, try to accomplish the following:

1. Import the NetAdapter module (available in the latest version of Windows, both client and server). Using the Get-NetAdapter cmdlet, display a list of non-virtual network adapters (that is, adapters whose Virtual property is False, which Powershell represents with the special $False constant).

   import-module NetAdapter
   get-netadatper -physical

2. Import the DnsClient module (available in the latest version of Windows, both client and server). using the Get-DnsClientCache cmdlet, display a list of A and AAAA records from the cache. Hint: if your cache comes up empty, try visiting a few Web pages first to force some items into the cache.

   Import-Module DnsClient
   Get-DnsClientCache -type AAAA,A

3. Display a list of hotfixes that are security updates

   Get-Hotfix -Description 'Security Update'

4. Using Get-Service, is it possible to display a list of services that have a start type of Automatic, but that aren't currently stated?

   No. the object that Get-Service uses doesn't have that much information. We would need to use WMI and the Win32_Service class.

5. Display a list of hotfixes that were installed by the Administrator, and which are updates. Note that some hotfixes won't have an "installed by" value - that's okay.

   get-hotfix -Description | Update where {$_.InstalledBy -match "administrator"}

6. Display a list of all the processes running as either Conhost or Svhost.

   get-process -name svchost, conhost