Landesberufsschule 4 Salzburg

Übungen im

IT-Laboratorium

Powershell

für die Übung Nr. 2

Katalog - Nr.: ***1***

Name : ***Valentin Adlgasser***

Jahrgang : ***2020***

Datum der Übung : ***25.05.2020***

Inhalt

1. Anweisung der Übung:

Testen Sie Powershell-Befehle und beschreiben Sie was dieses tun.

2. Einleitung

Powershell ist ein Framework von Microsoft zum automatisieren und verwalten von Systemen. Zum benutzen von Powershell stehet ein Kommandozeileninterpreter und eine Skriptsprache zur Verfügung. Powershell ist außerdem plattformübergreifend nutzbar.

3. Inventarliste

Heimrechner

4. Übungsdurchführung

a. Get-help

Der get-help Befehl, liefert eine Übersicht aller im Parameter angegeben Befehle. Im Folgenden eine Übersicht aller Funktionen die das Wort Computer enthalten.

```
PS C:\Users\Valentin> get-help -name computer

Name                                Category  Module                    Synopsis
----                                --------  ------                    --------
Add-Computer                        Cmdlet    Microsoft.PowerShell.M... ...
Checkpoint-Computer                 Cmdlet    Microsoft.PowerShell.M... ...
Disable-ComputerRestore             Cmdlet    Microsoft.PowerShell.M... ...
Enable-ComputerRestore              Cmdlet    Microsoft.PowerShell.M... ...
Get-ComputerInfo                    Cmdlet    Microsoft.PowerShell.M... ...
Get-ComputerRestorePoint            Cmdlet    Microsoft.PowerShell.M... ...
Remove-Computer                     Cmdlet    Microsoft.PowerShell.M... ...
Rename-Computer                     Cmdlet    Microsoft.PowerShell.M... ...
Reset-ComputerMachinePassword       Cmdlet    Microsoft.PowerShell.M... ...
Restart-Computer                    Cmdlet    Microsoft.PowerShell.M... ...
Restore-Computer                    Cmdlet    Microsoft.PowerShell.M... ...
Stop-Computer                       Cmdlet    Microsoft.PowerShell.M... ...
Test-ComputerSecureChannel          Cmdlet    Microsoft.PowerShell.M... ...
Get-MpComputerStatus                Function  Defender                  ...
```

b. New-PSDrive

Mit diesem Befehl kann man einen neuen Powershell-Drive erstellen. Hier habe ich einen neuen Drive im Laufwerk D:\Test mit dem Name X erstellt.

Der Parameter „PSProvider“ gibt an welchen Provider Powershell verwenden soll. Wenn man auf das Filesystem zugreift benutz man „Filesystem“, für RegKeys würde man „Registry“ verwenden. Die anderen Parameter sind, meiner Meinung nach, selbsterklärend.

```
PS C:\Users\Valentin> New-PSDrive -Name "X" -PSProvider "FileSystem" -Root "D:\Test"

Name           Used (GB)     Free (GB) Provider      Root
----           ---------     --------- --------      ----
X                   0,00        607,30 FileSystem    D:\Test
```

c. Get-Childitem

Der Get-Childitem Befehl zeigt alle Dateien und Ordner in einem spezifizierten Ordner.

In der Übung habe ich nach allen .exe- und .txt-Dateien in den Ordner C:\ und C:\Windows gesucht. Der Path-Parameter gibt die zu durchsuchenden Ordner an, ich habe danach den Include-Parameter benutzt, um nach Dateiendungen zu suchen. Wenn man den Include-Parameter benutzt muss man am Ende des Paths einen „*“ machen.

```
PS C:\Users\Valentin> Get-Childitem -Path C:\*, C:\Windows\* -Include *.exe, *.txt


    Verzeichnis: C:\Windows


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        15.09.2018     09:28          78848 bfsvc.exe
-a----        13.05.2020     08:50        4443368 explorer.exe
-a----        10.03.2020     19:47        1071616 HelpPane.exe
-a----        15.09.2018     09:29          18432 hh.exe
-a----        04.05.2019     16:14         254464 notepad.exe
-a----        09.04.2019     18:43         358400 regedit.exe
-a----        29.11.2019     14:14         132608 splwow64.exe
-a----        15.09.2018     09:29          11776 winhlp32.exe
-a----        15.09.2018     09:29          11264 write.exe
```

d. Copy-Item

Copy-Item macht genau das, was man erwartet. Es kopiert Dateien.
Wir sollten alle Dateien, die wir in der vorherigen Übung gefunden haben in das Verzeichnis aus Übung 2 kopieren.
Die Parameter sind selbsterklärend.

```
PS C:\Users\Valentin> Copy-Item -Path C:\*, C:\Windows\* -Include *.exe, *.txt -Destination D:\Test
```

e. Get-Childitem sortiert

Man kann die Tabelle, die bei Get-Childitem erstellt wird, natürlich sortieren. Dazu benutzt man den Pipe-Operator und danach das CMDlet format-list oder format-table.
In dieser Übung sortiere ich die Dateien nach Zugriffszeit, Name, Erstellungsdatum, Länge und Verzeichnis.

```
PS C:\Users\Valentin> Get-Childitem -Path C:\Windows\* -Include *.exe, *.txt, *.bin | format-table LastAccessTime, Name,
 CreationTime, Length, Directory

LastAccessTime      Name          CreationTime        Length Directory
--------------      ----          ------------        ------ ---------
15.09.2018 09:28:22 bfsvc.exe     15.09.2018 09:28:22   78848 C:\Windows
13.05.2020 08:50:34 explorer.exe  13.05.2020 08:50:34 4443368 C:\Windows
10.03.2020 19:47:34 HelpPane.exe  10.03.2020 19:47:34 1071616 C:\Windows
15.09.2018 09:29:18 hh.exe        15.09.2018 09:29:18   18432 C:\Windows
15.09.2018 09:28:57 mib.bin       15.09.2018 09:28:57   43131 C:\Windows
04.05.2019 16:14:40 notepad.exe   04.05.2019 16:14:40  254464 C:\Windows
09.04.2019 18:43:36 regedit.exe   09.04.2019 18:43:36  358400 C:\Windows
29.11.2019 14:14:57 splwow64.exe  29.11.2019 14:14:57  132608 C:\Windows
15.09.2018 09:29:27 winhlp32.exe  15.09.2018 09:29:27   11776 C:\Windows
15.09.2018 09:29:24 write.exe     15.09.2018 09:29:24   11264 C:\Windows
```

Und das ganze noch für ein anderes Verzeichnis:

```
PS C:\Users\Valentin> Get-Childitem -Path 'D:\SteamLibrary\steamapps\common\DiRT Rally 2.0\*' -Include *.exe, *.dll, *.b
in | format-table LastAccessTime, Name, CreationTime, Length, Directory

LastAccessTime      Name                  CreationTime        Length Directory
--------------      ----                  ------------        ------ ---------
02.07.2019 19:04:16 bink2w64.dll          02.07.2019 19:04:16   376832 D:\SteamLibrary\steamapps\common\DiRT Rally 2.0
02.07.2019 19:12:56 CrashRpt1405.dll      02.07.2019 19:12:56   216064 D:\SteamLibrary\steamapps\common\DiRT Rally 2.0
02.07.2019 19:04:16 CrashSender1405.exe   02.07.2019 19:04:16  1158656 D:\SteamLibrary\steamapps\common\DiRT Rally 2.0
02.07.2019 19:04:16 d3dcompiler_47.dll    02.07.2019 19:04:16  4173928 D:\SteamLibrary\steamapps\common\DiRT Rally 2.0
04.03.2020 09:51:48 dirtrally2.exe        04.03.2020 09:51:48 23996928 D:\SteamLibrary\steamapps\common\DiRT Rally 2.0
14.08.2019 17:54:42 openvr_api.dll        14.08.2019 17:54:42   593184 D:\SteamLibrary\steamapps\common\DiRT Rally 2.0
06.07.2019 16:37:04 steam_api64.dll       06.07.2019 16:37:04   250656 D:\SteamLibrary\steamapps\common\DiRT Rally 2.0
07.07.2019 18:15:12 WinPixEventRuntime.dll 07.07.2019 18:15:12    34920 D:\SteamLibrary\steamapps\common\DiRT Rally 2.0
```

f. Get-Process

Der Get-Process Befehl liefert alle derzeit ausgeführten Prozesse zurück.
In der Übung führe ich alle Prozesse, absteigend nach Prozessorzeit, mit den Attributen Name, Prozessorzeit, Firma, Threads und Startzeit. Außerdem zeige ich nur die ersten 10 Prozesse an.
Zum Anzeigen der ersten 10 Prozesse benutze ich den Select-Object Befehl, zum absteigenden sortieren den Sort-Object Befehl.
Parameter sind wieder selbsterklärend.

```
PS C:\Users\Valentin> get-process | Sort-Object -Descending TotalProcessorTime| Select-Object -first 10 | format-table N
ame, TotalProcessorTime, Company, Threads, StartTime

Name     TotalProcessorTime Company                Threads                          StartTime
----     ------------------ -------                -------                          ---------
chrome   00:09:47.4687500   Google LLC             {13552, 13764, 14076, 14080...} 25.05.2020 15:46:07
steam    00:03:43.2812500   Valve Corporation      {9708, 9020, 6400, 8076...}     25.05.2020 15:45:48
chrome   00:02:45.0625000   Google LLC             {6036, 9248, 9320, 9324...}     25.05.2020 15:46:06
Music.UI 00:01:51.7187500   Microsoft Corporation  {8496, 1688, 3832, 5612...}     25.05.2020 16:16:55
chrome   00:01:42.4531250   Google LLC             {13560, 13720, 13724, 13736...} 25.05.2020 15:46:07
WINWORD  00:01:24.7500000   Microsoft Corporation  {11436, 4420, 8332, 6096...}    25.05.2020 22:40:19
Teams    00:01:17.3906250   Microsoft Corporation  {11408, 11444, 11448, 11452...} 25.05.2020 15:45:55
explorer 00:00:55.7343750   Microsoft Corporation  {6912, 5924, 116, 2328...}      25.05.2020 15:45:31
Teams    00:00:47.3125000   Microsoft Corporation  {11248, 9860, 10020, 11736...}  25.05.2020 15:45:57
chrome   00:00:42.7187500   Google LLC             {13896, 13528, 12112, 12480...} 25.05.2020 15:46:20
```

g. Tabelle in CSV exportieren

Mit dem Befehl Export-CSV kann man eine Tabelle in eine CSV exportieren.
In der Übung habe ich genau das gemacht.

```
PS C:\Users\Valentin> Get-Childitem -Path C:\Windows\* -Include *.exe, *.ini |Sort-Object -Descending LastWriteTime | Fo
rmat-Table Name, Length, LastWriteTime | Export-CSV D:\Test\Test.csv
```

h. Where-Object

Mit Where-Object kann man Filter erstellen. Dieser Filter kann Vergleichsoperatoren oder logische Operatoren beinhalten und ist nicht schwer zu verstehen.

```
PS C:\Users\Valentin> Get-Childitem -Path C:\Windows\* -Include *.exe | Where-Object -Filterscript {$_.length -ge 25kb -
or $_.length -lt 17kb} | sort-object length -descending


    Verzeichnis: C:\Windows


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        13.05.2020     08:50        4443368 explorer.exe
-a----        10.03.2020     19:47        1071616 HelpPane.exe
-a----        09.04.2019     18:43         358400 regedit.exe
-a----        04.05.2019     16:14         254464 notepad.exe
-a----        29.11.2019     14:14         132608 splwow64.exe
-a----        15.09.2018     09:28          78848 bfsvc.exe
-a----        15.09.2018     09:29          11776 winhlp32.exe
-a----        15.09.2018     09:29          11264 write.exe


PS C:\Users\Valentin> Get-Childitem -Path C:\Windows\ | Where-Object -Filterscript {$_.length -ge 300kb -or $_.length -l
t 3000kb} | sort-object LastAccessTime -descending


    Verzeichnis: C:\Windows


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        26.05.2020     00:03                Prefetch
-a--s-        25.05.2020     22:38          67584 bootstat.dat
d-----        25.05.2020     20:14                Temp
-a----        25.05.2020     17:48            276 WindowsUpdate.log
d-----        25.05.2020     17:26                Logs
d-----        25.05.2020     15:49                System32
d-----        25.05.2020     15:46                INF
-a----        25.05.2020     15:42          28176 PFRO.log
d-r---        25.05.2020     13:32                Microsoft.NET
d-----        24.05.2020     22:27                AppReadiness
-a----        19.05.2020     19:36          11422 setupact.log
d-----        16.05.2020     13:03                LiveKernelReports
d-----        15.05.2020     16:47                WinSxS
d-r-s-        15.05.2020     16:46                assembly
d-----        14.05.2020     15:14                TextInput
d-----        14.05.2020     15:14                SysWOW64
d-----        14.05.2020     15:14                ShellExperiences
d-----        14.05.2020     15:14                PolicyDefinitions
d-----        14.05.2020     15:14                Provisioning
d-----        14.05.2020     15:14                bcastdvr
d-----        14.05.2020     07:59                CbsTemp
```

5. Einsatzgebiet

Powershell kann so gut wie überall verwendet werden. Meistens allerdings in Firmen, um Prozesse zu automatisieren und so Arbeitszeit einzusparen.

6. Erkenntnisse

Keine neuen Erkenntnisse, da ich Powershell jeden Tag in der Firma benutze