PowerShell

Process Management

Aufgabe 1

Run a PowerShell as administrator.

a. Use the Cmdlet *Get-Process* to retrieve a list of all running processes.

b. Use a pipe[1] to format and output the result set either as a table (Cmdlet: *Format-Table*) or as a list (Cmdlet: *Format-List*).

c. Use a pipe to page the result set (as with *more* in the command prompt). The corresponding Cmdlet it *Out-Host -Paging*[2].

d. Use the command *Get-Process* | *Get-Member* to get a list of all properties of a process managed by the operating system.

[1] Pipe is the technical term fort he character **|** , which in QWERTY keyboard layout is the third option for the key to the immediate left of ‚z‘. By chaining cmdlets with a pipe, the output of one Cmdlet can serve as input to the next Cmdlet in the pipeline.
[2] The option -*Paging* is a parameter to the Cmdlet *Out-Host*. A parameter defines the behavior of a program, command of Cmdlet more in detail.

Information

The Cmdlet *Where-Object* allows filtering results according to certain properties of the objects in the result set. Syntactically, the filter itself is provided as a parameter in curly brackets behind the name of the Cmdlet[3]. In this context the expression *$_* functions as a reference to each object that is passed into the Where-Object Cmdlet[4].

Example: Listing all *svchost.exe* processes using PowerShell.

Get-Process | *Where-Object { $_.Name -eq “svchost“ }*

We just need to know the name of the desired property that should be contained in the filter as well as a threshold value and the fitting comparison operator:

• *-eq*	equal
• *-ne*	not equal
• *-gt*	greater than
• *-lt*	lesser than
• *-ge*	greather than or equal
• *-le*	lesser than or equal
• *-like*	like for pattern matching in String objects

The above list is only an excerpt of the complete list of all comparison operators. Some others like -ceq, -match, -in, -contains, -notlike are also used in some of the exercises in the upcoming chapters. See the documentation for instructions and examples of how to use them correctly.

[3] All filter script blocks are technically predicate functions. These predicates return a Boolean value, i.e. *True* or *False* (in a logical sense). The logical *True* is call *$True* within PowerShell and the logical *False $False*.

[4] In programming *Where-Object* can be regarded as an equivalent to a loop (with an anonymous index variable) that contains a conditional If-statement.

Exercise 2

Use the Cmdlets *Get-Process* and *Where-Object* chained together in a pipeline.

- List all processes that control services:

  Property: SI (Session-ID), *Threshold*: 0 (for services)

- List all processes using more than 10 MB of RAM:

  Property: WorkingSet, *Threshold*: 10000000 (or 10 MB)

- List all processes that have consumed at least 0.5 seconds of CPU time.

  Property: CPU, *Threshold*: 0.5

- (Bonus) Find the number[5] of processes with name *svchost.exe*.

  Property: Name, *Threshold*: *svchost*

- (Bonus) Determine how many properties an object of type *System.Diagnostics.Process* possess according to the output of *Get-Member*.

Exercise 3 (Bonus)

Besides the comparison operators PowerShell also implements the basic logic operators:

- *-And* for the logical ‚and‘
- *-Or* for the logical (inclusive) ‚or‘
- *-Not* for the logical ‚not‘
- *-Xor* for the logical exclusive ‚or‘

a. Use the Cmdlets *Get-Process* and *Where-Object* chained together in a pipe.

- List all processes whose name begins with the letter ‚S‘:

  Property: ProcessName, *Threshold*: S*

- List all processes whose name begins with the letter ‚S‘ or with the letter ‚I‘ (which sort of ‚or‘ are you using here?:

  Property: ProcessName, *Thresholds*: S* or I*

- List all processes whose name does not begin with the letter ‚S‘:

  Property: ProcessName, *Threshold*: S*

[5] Use the Cmdlet *Measure-Object* at the end of the pipeline.

b. Filter the output of the command *Get-ChildItem C:\Windows -File* with the Cmdlet *Where-Object*.
   - List all filed that are larger than 5 Kilobyte (kB), but smaller than 100 Kilobyte (kB).

     Property: Length, *Thresholds*: 5kB[6] and 100kB
   - List all files that are smaller than 10 kB or whose name begins with *win*.

     Property: Length und Name, *Threshold*: 10kB and win*
   - List all files that are either smaller than 10 kB or whose name begins with *win*. Compared to the command above: which files are missing in the output of this command and why?

Exercise 4

Start an instance of the editor *Notepad*[7].

Get the correspoding process object with *Get-Process -Name notepad*.

Stop the process using the Cmdlet *Stop-Process* by giving the process name as a parameter[8].

Use *Start-Process* to start a new instance of *Notepad*.

Stop this process using *Stop-Process* by giving the process ID as a parameter[9].

Exercise 5

a. Execute the command *Get-Command -Noun Compute*r to find available Cmdlets related to the object *computer*.
b. Note down the Cmdlet for rebooting a computers and test it on your machine.
c. Search for the Cmdlet to shut down a computer. Execute this Cmdlet with the parameter *-WhatIf*. Describe the functionality of the risk mitigation parameter.

[6] In PowerShell the abbreviations kB (Kilobyte), MB (Megabyte) and GB (Gigabyte) for the sizes of objects can be directly used.
[7] Under GNU/Linux choose the default text editor of the distribution.
[8] The parameter *-Name* is optional, because it is a positional parameter
[9] The parameter *-Id* is optional, because it is a positional parameter.