Software

PowerShell

- Powershell 5 - alte original MS Powershell (in Windows Desktop und Server, im Startmenue oder Aufruf "powershell")
- Powershell 7.1 - aktuelle quelloffene Powershell Core, muß nachgeladen werden im MS Store oder bei Github, Aufruf mit "pwsh".
  basiert auf (quelloffener) .NET Plattform
  läuft unter MacOS, Linux und Windows
  als Script-Editor/Debugger dient MS Visual Studio Code

Powershell (5) ISE als Admin starten! (aus der Powershell heraus: "ise")
- Script-Ausführung dauerhaft aktivieren: Set-ExecutionPolicy RemoteSigned
 (nur noch PS1 aus dem Internet oder ohne vertraute Signatur werden geblockt)
  Mehr Infos mit help about_Execution_Policies (Bypass / Restricted)
- Script remote ausführen: Enable-PSRemoting

- $PSVersionTable - Version
- update-help -> zuerst ausführen! (aktualisiert die Hilfe, dauert Minuten)
- get-help -> Hilfe allgemein
- help Befehl -> Hilfe zum Befehl
- help Befehl -detailed -> detaillierte Hilfe zum Befehl
- help about_* -> Hilfe zum PS-Konzept

- get-command -> Liste aller Befehle
- # - Kommentarzeile
- <# ... #> - Kommentarblock, auch mehrzeilig

- get-(befehl) -example -> zeigt Beispiele

In der Powershell lassen sich alle externen .exe-Kommandos wie robocopy oder bcdedit, netsh, notepad usw. aufrufen.
Die meisten CMD-Befehle wie cd, md, dir, del funktionieren auch.

Viele weitere PS-Module in der MS PowerShell Gallery.

Ausgabebefehle:
- Write-Host - (auch farbig oder mit formatierten Arrays möglich)
- Write-Output (echo) - einfache Ausgabe
- Write-Error - gibt Fehlermeldungen aus

Add-On Menue / Webseite mit AddOns öffnen: praktische AddOns!
  - MS Script Browser: durchsucht Poswershell Scripte nach Stichworten

Beispiele
Get-Prozess - Liste aller Prozesse. (Parameter "*edge" schränkt auf Edge-Prozesse ein)
Get-Member - liefert Bestandteile eines vorher genannten Objektes.

(c) 2024 Uwe Kernchen <news@uwe-kernchen.de> | 2024-04-26 17:15
URL: https://uwe-kernchen.de/phpmyfaq/index.php?action=faq&cat=1&id=146&artlang=de
(C) https://uwe-kernchen.de

Software

Get-Prozess firefox | Get-Member - liefert alle verfügbaren Attribute
Format-List oder Format-Table - liefert formatierte Felder des vorher genannten Objektes
Get-Process firefox | Format-List Name, WorkingSet - liefert Speicherverbrauch aller Firefox-Instanzen
Get-Process firefox | Measure WorkingSet -Sum - liefert die Summe des Speicherverbrauch aller Firefox-Instanzen

Get-Service (gsv) - Liste aller installierten Services. (mit Status)
Where - Filter, kann mit "?" abgekürzt werden
-eq prüft auf Gleichheit (ne - ungleich, -gt - größer als, -lt - kleiner als, -ge -größer/gleich, -le - kleiner/gleich)
Get-Service | Where Status -eq Running - alle laufenden Dienste
Get-Service veeam* | Stop-Service - stoppt alle Dienste, die mit Veeam beginnen

Sort-Objekt (sort) - sortiert das übergebene Objekt
dir | sort Length -descending

Get-Date - Datum mit allen Objekten (Tag, Monat, Stunde, Tag des Jahres, Tag der Woche...)
New-Timespan - zum Berechnen von Tagesdifferenzen

Variablen müssen nicht definiert werden.
Groß- und Kleinbuchstaben werden nicht unterschieden.
Jedes PS-Fenster hat seine eigenen Variablen, die beim Schließen des Fensters gelöscht werden.
$antwort = 5 * 3

$1woche = (Get-Date) - (New-TimeSpan -Days 7)
$kannweg = dir *.log | WHERE LastWriteTime -lt $1woche
Liefert Liste aller Log-Dateien > 1 Woche

Die Variable kann wieder in eine Filter-Pipeline übergeben werden.
$platz = $kannweg | measure Length -Sum

Berechnung mit der Eigenschaft "Sum" des Objektes "$platz".
$MByte = $platz.Sum / (1024*1024)

Mit dieser Methode kann auf alle Objekte zugegriffen werden.
$Jahr = (Get-Date).Year

Remove-Item (del) - löscht Datei
$kannweg | Remove-Item - löscht die oben definierte Liste in $kannweg

ForEach.Object (foreach oder %) - Schleife
$kannweg | % { move $_ e:\temp\test } - verschiebt Dateien aus "$kannweg" nach e:\temp\test
{} - umschließt Script-Block
Ruft nacheinander für jeden Eintrag von "$kannweg" den Script-Block auf.

(c) 2024 Uwe Kernchen <news@uwe-kernchen.de> | 2024-04-26 17:15
URL: https://uwe-kernchen.de/phpmyfaq/index.php?action=faq&cat=1&id=146&artlang=de
(C) https://uwe-kernchen.de

Zuvor speichert es den aktuellen Eintrag von $kannweg in die Variable "$_", die als Platzhalter genutzt werden kann.

Auch WHERE nimmt Script-Block entgegen.
Get-Process | where {$_.Company -match 'Microsoft'} oder
Get-Process | where {$_.Company -match 'Microsoft' -and $_.WS -ge 10*1024*1024} - liefert Prozesse vom MS > 10MB

Die Powershell bildet REG-Schlüssel, Zertifikatsspeicher, Umgebungsvariablen als PSDrives ab.
Get-PSDrive - Liste aller PSDrives.
cd "HKCU:\Control Panel\Desktop\" - wechselt in den REG-Schlüssel, Anzeige mit "dir"

Get-Item-Property (gp) - liest Eigenschaften des angegebenen Objektes
gp . WallPaper - liefert Dateinamen des Bildschirmschoners.
Set-Item-Property (sp) - schreibt Eigenschaften des angegebenen Objektes
Remove-Item-Property (rp) - löscht Eigenschaften des angegebenen Objektes
help Registry - liefert Liste aller zur REG-Bearbeitung interessanten Befehle

get-childitem -path hklm:\software - Softwareliste
(get-childitem -path hklm:\software).Property - Softwareliste gefiltert

cd "Env:" - wechselt in Umgebungsvariablen. Hier geht wieder dir, gp, sp, rp...
$env:path - Inhalt von Umgebungsvariable %PATH%
$env:temp - Inhalt von Umgebungsvariable %TEMP%

Kontrollstrukturen:
(While, Do-While, Do-Until, For, Foreach)
if (Bedingung)
 { Befehle }
elseif (Bedingung)
 { Befehle }
else
 { Befehle }
IF (Test-Path "$env:TEMP\error.log") ... - prüft die Existenz eines Datei- oder Ordnernamens

Funktionen:
function NAME
{ bla, bla }

Aufruf einfach mit "NAME".
Funktionen können Parameter übergeben bekommen, für die man Defaults setzen kann.
Rückgabewerte per "return WERT" - beendet Schleife und Funktion sofort

```
function LogsLöschen($Ordner = ".", $Alter = 7)
{
```


(c) 2024 Uwe Kernchen <news@uwe-kernchen.de> | 2024-04-26 17:15
URL: https://uwe-kernchen.de/phpmyfaq/index.php?action=faq&cat=1&id=146&artlang=de
(C) https://uwe-kernchen.de

```
 $datum = (Get-Date) - (New-TimeSpan -Days $Alter)
 dir "$Ordner\*.log" | ? LastWriteTime -lt $datum | Remove-Item
}
```

Aufruf ist möglich mit oder ohne Parameter, auch nur Parameter2 (Bsp. 4):

```
LogsLoeschen

LogsLoeschen Temp
LogsLoeschen Temp 30
LogsLoeschen -Alter 14
```

Beispielfunktion mit Array/Liste und Return, formatiert Zahlen übersichtlich und mit Einheit.
- return "{0:f2} - erstes Argument (0) wird als Gleitkommazahl mit 2 Nachkommastellen ausgegeben
- -f $Wert, $units[$unit] - formatierte Ausgabe

```
function BinFormat($Wert)
{
 $units = "", "Kilo", "Mega", "Giga", "Tera", "Peta", "Exa", "Zetta",
"Yotta"
 $unit = 0
 while($Wert -ge 1024 -and $unit -lt $units.Length - 1)
 {
  $Wert = $Wert / 1024
  $unit++
 }
 return "{0:f2} {1}" -f $Wert, $units[$unit]
}
```

Externe PS-Scripts lassen sich einbinden mit ". MeineFunktionen.ps1".

Alle beim PS-Start per Default ausgeführtenScripts erfährt man mit:

```
$profile | fl * -Force
```

Dabei gibt es Scripts für AllUser und CurrentUser, die entsprechend ergänzt werden können.
"CurrentHost" ist nur die aktuelle PS-Sitzung.

(c) 2024 Uwe Kernchen <news@uwe-kernchen.de> | 2024-04-26 17:15
URL: https://uwe-kernchen.de/phpmyfaq/index.php?action=faq&cat=1&id=146&artlang=de
(C) https://uwe-kernchen.de

Beispielscript mit .NET Framework Einbindung.
Wenn der aktuelle User ein Admin ist, färbe Fenster rot.

```
function BinIchAdmin() {
 $identity = [System.Security.Principal.WindowsIdentity]::GetCurrent()
 $princ = New-
Object System.Security.Principal.WindowsPrincipal($identity)
 return $princ.IsInRole([System.Security.Principal.WindowsBuiltInRole]
::Administrator)
} & {
 $ui = (Get-Host).UI.RawUI
 if(BinIchAdmin) {
  $ui.BackgroundColor = "DarkRed"
  Clear-Host
 }
}
```

Reguläre Ausdrücke
Bsp.: Liest Datei test.log und gibt nur Zeilen aus, in denen "Error" steht.

```
Select-String test.log 'Error'
```

Bsp2: Suchmuster "Error" **oder** "Warning". (Pipe steht für ODER)

```
Select-String test.log 'Error|Warning'
```

Bsp3: Aufruf geht auch mit Wildcards (*.log) und liest dann alle betroffenen Dateien.

Der Vergleichsoperator **-match** dient zum filtern.
Beispiel:
- "DIR" landet über die Pipe bei WHERE
- "-match" liefert nur WAHR, wenn der folgende Ausdruck passt
- "." (Punkt) steht normalerweise für ein beliebiges Zeichen, ähnlich Fragezeichen in Dateimasken (soll er hier aber nicht)
- "\" Escape-Zeichen sorgt dafür, dass das folgende Zeichen (Punkt) für sich selbst steht
- abschließendes "$" pinnt den Such-String an das Ende des durchsuchten Textes.
  (beginnt der RegEx mit "^", wird die Zeichenfolge am Anfang des Textes gesucht)
Das Beispiel sucht also Dateinamen, die mit ".txt" enden.

```
dir | Where Name -match '\.txt$'
```


(c) 2024 Uwe Kernchen <news@uwe-kernchen.de> | 2024-04-26 17:15
URL: https://uwe-kernchen.de/phpmyfaq/index.php?action=faq&cat=1&id=146&artlang=de
(C) https://uwe-kernchen.de

RegEx Syntax kennt auch Zeichenklassen:
'[0-9]' - beliebige Ziffer
'[a-z]' - beliebigen Buchstaben (KEINE Umlaute!)
'[a-z0-9_]' - alle Ziffern, Buchstaben und der Tiefstrich
'h[aiu]t' - findet "hat", "hit" und "hut", aber nicht "haut"
'[^a-z]' - alle Zeichen **ausser** a-z
'\d' - Dezimalziffern
'\w' - Wortzeichen, also Zahlen, Umlaute und Buchstaben
'\s' - Wortzwischenräume (Leerzeichen, Tab, Umbruch)
Großschreibung kehrt Bedeutung um, also '\D' - alles ausser Ziffern
'.' - beliebige Zeichen
Weitere Syntax und Beispiele in CT 08/2018 S. 172ff.

Paketmanagement
- zuerst Modul (als Admin) aktualisieren:

```
Install-Module PackageManagement -Force
Install-PackageProvider NuGet -Force
Install-PackageProvider PowerShellGet -Force
```

- Übersicht aller Funktionen:
  (gcm) Get-Command -Module PowerShellGet
- gewünschtes Modul installieren: Install-Module ... (-Scope CurrentUser installiert in Userprofil)
- GUI zur Modulsuche und Auswahl:

```
Find-Module -Tag Utilities | Out-Gridview -PassThru | Install-
Module -Scope CurrentUser
```

Beispiel: **eigenen Host, IP, Username ermitteln**
- Comutername über Enviroment-Variable: "$env:computername" oder "get-childitem -path env:computername"
- IP ermitteln: hier
- Username: $Env:USERNAME

Computerinfo ermitteln:
(ausführlich Windows, Sprache, 32/64-bit, WoL, BIOS, Hardware)
Powershell: get-computerinfo
einschränken auf Windows Versionsinformation:
Powershell: get-computerinfo | select windows* (oder OsVersion)

Quellen und Links:

- administrator.de : Links und Leitfäden

(c) 2024 Uwe Kernchen <news@uwe-kernchen.de> | 2024-04-26 17:15
URL: https://uwe-kernchen.de/phpmyfaq/index.php?action=faq&cat=1&id=146&artlang=de
(C) https://uwe-kernchen.de

Software

- CT 2/2018 S.166 ff, CT 6/2018 S.168 ff, CT 8/2018 S.172 ff, CT 12/2018 S. 174 ff, CT 25/2021 S. 178 ff.
- Powershell Leitfaden für Anfänger: https://docs.microsoft.com/de-de/powershell/scripting/learn/ps101/01-getting-started?view=powershell-7

Eindeutige ID: #1145
Verfasser: Uwe Kernchen
Letzte Änderung: 2023-05-29 11:56

(c) 2024 Uwe Kernchen <news@uwe-kernchen.de> | 2024-04-26 17:15
URL: https://uwe-kernchen.de/phpmyfaq/index.php?action=faq&cat=1&id=146&artlang=de
(C) https://uwe-kernchen.de