Interaktiver Modus

Commandlet-Aufbau

`Verb-Noun`	**Verb:** Get, Add, Copy, Set, ... **Noun:** Process, Item, Help, ...

Hilfefunktionen

`Update-Help`	Als Admin: aktualisiert Hilfedateien
`Get-Help <cmd>`	Zeigt Hilfe zu Cmdlet
`Get-Help *Item`	Zeigt alle Cmdlets, die mit Item enden
`Get-Command <cmd>`	Hilfe zu Commands (`-Verb Get` holt alle Get-Cmdlets)
`Get-Help about_*`	Zeigt alle About-Docs (z. B. `about_if`).
`<var> | Get-Member`	Variablen und Commandlets untersuchen
`<Cmd> -WhatIf`	Bei kritischen Aktionen Ausführung emulieren

Standardkanäle

`<a> i> <file>`	Leitet Stream i von a in $file$
`<a> i>&j`	Leitet Stream i von a in Stream j

Aliase

`New-Alias`	Erstellt neuen Alias
`Remove-Alias`	Löscht einen Alias
`Get-Alias -Definition <cmd>`	Zeigt alle Aliase zu einem Commandlet

Nr	PowerShell	Linux	Nr	PowerShell
0		`stdin`	4	`verbose`
1	`sucess`	`stdout`	5	`debug`
2	`error`	`stderr`	6	`information`
3	`warning`		*	`all streams`

Dateisystem

Ordner wechseln, Items anlegen, copieren, löschen, Infos holen, pwd, tree?, find?

`Set-Location`	In Ordner wechseln
`Get-Location`	Aktuellen Ornerpfad holen
`New-Item`	Erstellt Datei oder Ordner
`Copy-Item`	Kopiert Datei oder Ordner
`Remove-Item`	Löscht Datei oder Ordner
`Get-Item`	Holt Meta-Informationen eines Items (z. B. Datei) ein
`Get-Content`	Liest Inhalt einer Datei ein
`Get-ChildItem`	Holt ein Item und seine Kinder-Items (Unterordner)
`Tree`	Holt ein Item und seine Kinder-Items (Unterordner)
`Find`	Holt ein Item und seine Kinder-Items (Unterordner)

Nützliche Commandlets (kleine Auswahl)

`Get-Date`	Holt das aktuelle Datum
`Write-Host`	Erzeugt eine Ausgabe auf `stdout`
`Write-Debug 'msg' -Debug`	Schreibt eine Debugnachricht und aktiviert den Debug-Modus

Pipelining

`<a> | <b>`	Leitet `stdout` von a in `stdin` von b
`Where-Object{ }`	Filtert Objekte basierend auf einer Bedingung.
`Select-Object`	Wählt bestimmte Eigenschaften eines Objekts aus.
`Sort-Object`	Sortiert Objekte nach einem bestimmten Kriterium.
`Foreach-Object{ }`	Anweisungen pro Objekt ausführen
`Group-Object`	Gruppiert anhand einer Eigenschaft der Objekte.
`Get-Member`	Metadaten zu Objekt ausgeben
`Measure-Object`	Min, Max, Sum, Avg
`Compare-Object`	Zwei Objektmengen vergleichen
`Format-List`	Ausgabe formatieren (viele Format-Varianten)
`Tee-Object <a> | <b>`	Splittet `stdout` in a und b auf
`Get-Help about_Pipelines`	Hilfeartikel zu Pipelines

Skripting

Parameter in Skripten

`./script.ps1 <arg1> [, ..., <argN>]`	
`$args.Count`	Anzahl der Argumente prüfen
`$args.[i]`	**Positionale** Argumente an Stelle i auslesen
`./script.ps1 -par1 <w> [, ..., -parN <w>]`	
`param([typ]$par1, ... [typ]$parN)`	**Benannte** Parameter mit Typ definieren.

Umgang mit Variablen

`[int] $x = 5`	Zuweisung einer typisierten Variablen (Typ ist optional)
`[int] $x = "3.45" -as [Int]`	Konvertiert einen String-Wert in Int und schreibt ihn nach $x
`$x.GetType()`	Liefert Typinfos von $x
`$x.GetType(). FullName`	Liefert Typnamen von $x
`Clear-Variable x`	Löscht **Inhalt** von $x
`Remove-Variable x`	Löscht **Deklaration** von $x
`$true $false`	Wahr/falsch
`$Home`	Home-Folder des Nutzers
`$PSHome`	Installationsordner von PS
`$Error`	Liste aller Fehler seit Start der PowerShell
`Get-Item Variable:H*`	Zeigt alle definierten Variablen an, die mit H beginnen
`$x | Get-Member`	Zeigt Typ, Member, Methoden zu der Variablen an
`Get-Help about_Variables`	Hilfeartikel zu Variablen

Umgang mit Strings

`"Hi"` bzw. `'Hi'` bzw. `@'Hi@'`	Versch. Stringliterale (@-Notation: "Here-String")
`"a"+ $x + "c"`	Konkatenation
`"PC $nr"`	Ausdruckauflösung
`"Date: $(Get-Date)"`	Ausdruckauflösung
`"x:\$($pc)_.VHD.vhdx"`	Ausdruckauflösung
`$a.Substring(4,3)`	Textteil extrahieren $[5, 7]$, fängt bei Index 1 an
`$myArr = $x -Split "<del>"`	Splitten String am Delimiter <del> auf
`$x = $myArr -Join"<del>"`	Verbindet Teilstringe aus myArr in x
`$x.replace("ü", "ue")`	Case-Sensitives Ersetzten von Teilstrings
`$x -replace "\bÖ", "Oe"`	Ersetzten von Teilstrings m.H. von regulären Ausdrücken
`"" | Get-Member -MemberType Method`	String-Methoden ansehen

Ein- und Ausgabe

Hier auch reinschreiben:
`$x = Read-Host "x eingeben"`
Clear-Host

`New-Item`	Erstellt eine neue Datei oder einen neuen Ordner.
`Remove-Item`	Löscht eine Datei oder einen Ordner.
`Copy-Item`	Kopiert eine Datei oder einen Ordner.

Umgang mit nicht definierten Variablen

`$x ??= "n/a"`	Nimm x falls definiert, ansonsten schreibe Standardwert hinein
`${x}?.Property`	Wähle Property aus, falls existent, ansonsten null zurückgeben
`${arr}[100]`	Falls `arr` nicht existiert, gib null zurück

Arrays

`$x = "a","b","c"`	Array definieren
`$x = @(1,2,3)`	Array definieren
`$x = 1..10`	Zahlen von 1 bis 10 in x schreiben
`$x.Count`	Anzahl der Elemente holen
`$z = $x + $y`	Zwei Arrays verbinden
`$x = ("a", "b"), ("c", "d")`	Zwei-dimensionales Array erzeugen
`$x[0][1]`	Element "b"an $(0, 1)$ holen
`$x = @{a = "w1"; b = "w2"}`	Assoziatives Array erzeugen
`$x["a"]` bzw. `$x.a`	Wert von Index "a"auslesen