

# C\_Arbeit mit Dateisystem

Donnerstag, 19. März 2020 08:00

## 1 Arbeiten im Dateisystem

Das Arbeiten mit dem Dateisystem kann für Systemintegratoren eine wichtige Aufgabe sein. Nachfolgende Tabelle vermittelt einen ersten Eindruck über elementare Commandlets für die Arbeit mit dem Dateisystem in Powershell.

**Tabelle 31.1:** Wichtige Commandlets für die Arbeit mit dem Dateisystem

PowerShell-Commandlet	PowerShell-Alias	Befehl Klassische Windows-Kommandozeile	Befehl Unix „sh“	Beschreibung
Clear-Item	Cl i	–	–	Inhalt leeren
Copy-Item	Cpi, cpp, cp, copy	Copy	Cp	Kopieren von Elementen
Get-Content	Gc	Type	Cat	Holt den Inhalt
Get-Location	Gl, pwd	Pwd	Pwd	Holt das aktuelle Verzeichnis
Move-Item	Mi, move, mv, mi	Move	Mv	Bewegen von Elementen
New-Item	Ni (Funktion md)	–	–	Element anlegen
Remove-Item	Ri, rp, rm, rmdir, del, erase, rd	del, rd	rm, rmdir	Löschen von Elementen
Rename-Item	Rni, ren	Rn	Ren	Umbenennen eines Elements
Set-Content	Sc	(Umleitungen >)	(Umleitungen >)	Festlegen des Inhalts
Set-Item	Si	–	–	Inhalt festlegen
Set-Location	Sl, cd, chdir	cd, chdir	cd, chdir	Setzt das aktuelle Verzeichnis

Test-Path

### Arbeitsauftrag 1

Erläutern Sie das Cmdlet Get-Item (oben nicht aufgeführt). Worin besteht der Unterschied zu Get-Childitem?

### Arbeitsauftrag 2

Bereiten Sie eine kurze Präsentation vor, in der Sie Ihre Mitschüler die von Ihnen gewählte Funktion kurz erklären. Erstellen Sie hierzu eine Powerpointfolie, die die wesentlichen Informationen zu Ihrem Thema zusammenfasst. Überlegen Sie sich zudem eine kleine Übungsaufgabe. Bearbeitungszeit: 35 Minuten, Präsentations- und Übungszeit: 5-10 Minuten. Nutzen Sie für Ihre Recherche folgende Internetseite:

<http://technet.microsoft.com/de-de/scriptcenter/dd772285.aspx>

**Folgende Funktionen stehen zur Auswahl:**

[Copy Files or Folders](#)

[Create a New File or Folder](#)

[Delete a File or Folder \(Or Other Objects\)](#)

[Move a File or Folder](#)

[Rename a File or Folder](#)

[Retrieve a Specific Item](#)

[Verify the Existence of a File or Folder](#)

```
get-childitem $env:windir  
get-item $env:windir
```

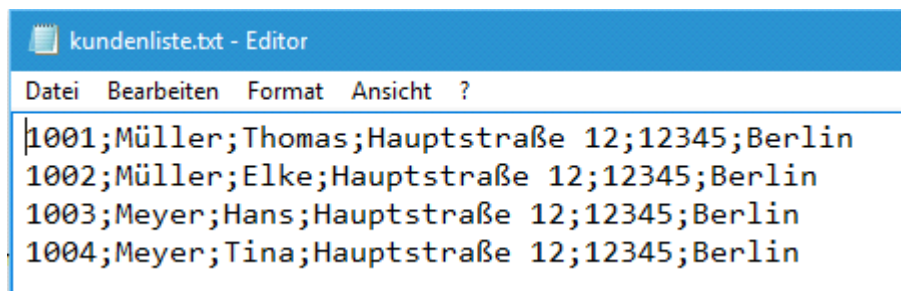
### **Arbeitsauftrag (Alle Aufgaben sind als Funktion zu codieren)!**

1. Löschen Sie eine von Ihnen zuvor erstellte Datei Ihrer Wahl.
2. Erstellen Sie einen Ordner mit mehreren Textdateien mit unterschiedlichen, frei wählbaren Inhalten. Speichern Sie auch eine Textdatei ohne Inhalt. Schreiben Sie anschließend ein Skript, das alle Textdateien löscht, deren Größe gleich 0KB ist.
3. Prüfen Sie anhand eines Skripts, ob es im Verzeichnis von Aufgabe 2 eine Datei mit dem Namen "report.txt" gibt. Sofern nicht, legen Sie diese an. Sofern diese bereits existiert, löschen Sie diese und legen Sie die Datei danach dann "frisch" neu an.
4. Kopieren Sie alle vorhandenen Dateien mit der Endung „.txt“ aus dem aktuellen Verzeichnis in ein Unterverzeichnis mit dem Namen „Textdateien“. Prüfen Sie auch hier wiederum, ob das entsprechende Unterverzeichnis bereits existiert und legen Sie dieses bei Bedarf zuvor noch an.
5. Erstellen Sie ein Backup-Skript, das bei jeder Ausführung einen Ordner erstellt, der das aktuelle Datum als Bestandteil seines Namens trägt. Lassen Sie im Rahmen dieses Skriptes alle Inhalte eines bestehenden anderen Ordners Ihrer Wahl dort als Kopie einfügen.
6. Stellen Sie sich folgendes Szenario vor: In einer Abteilung wird gemeinschaftlich ein Scanner genutzt. Der netzwerktaugliche Scanner speichert alle gescannten Dokumente in einem speziellen Ordner auf einem Netzwerklaufwerk ab. Unglücklicherweise vergessen verschiedene Kollegen, ihre Scanunterlagen nach der Nutzung zu löschen. Ihre Aufgabe ist es, ein Skript zu schreiben, mit dessen Hilfe alle Dateien in diesem Ordner gelöscht werden, die älter als 10 Tage sind. Legen Sie zu diesem Zweck einen Ordner mit einer passenden Anzahl von Dokumenten an und schreiben Sie ein Skript, das die gewünschte Bereinigung vornimmt.
7. Ein Kollege hat sich mächtig über Ihr eigenmächtiges Vorgehen beschwert. Nach heftigen Diskussionen haben Sie vereinbart, vor dem Löschen dieser Datei, diese zu kopieren und in einen Archivierungsordner zu verschieben. Passen Sie Ihr Skript entsprechend an.
8. Erweitern Sie das Skript, so dass der Inhalt des neu entstandenen Archivierungsordners jeweils am 31.12 eines Jahres gelöscht wird. Falls das aktuelle Datum nicht der 31.12. des Jahres ist, soll eine entsprechende Meldung erfolgen. Die Meldung erfolgt mithilfe der Windows Message-Box. Informieren Sie sich wie diese eingebunden werden kann.

## 1 Arbeiten mit Dateien

### 1. Eine Text-Datei auslesen

1. Erstellen Sie eine Textdatei mit dem Namen „kundenliste“. Zu jedem Kunden sollen folgende Informationen pro Zeile in folgender Reihenfolge erfasst werden: Kundennummer, Nachname, Vorname, Straße, PLZ, Ort. Jeder Eintrag soll durch ein Semikolon getrennt sein. Keine Leerzeichen dazwischen.



```
kundenliste.txt - Editor
Datei Bearbeiten Format Ansicht ?
1001;Müller;Thomas;Hauptstraße 12;12345;Berlin
1002;Müller;Elke;Hauptstraße 12;12345;Berlin
1003;Meyer;Hans;Hauptstraße 12;12345;Berlin
1004;Meyer;Tina;Hauptstraße 12;12345;Berlin
```

2. Lesen Sie die oben erstellte txt-Datei mit einem geeigneten Cmdlet aus und speichern Sie den kompletten Inhalt der Textdatei im Arbeitsspeicher. Lassen Sie sich diesen Arbeitsspeicherinhalt in der Konsole ausgeben. Nutzen Sie eine geeignete Methode (was war gleich nochmal eine Methode?) um den Typ des Speicherkonstrukts herauszufinden. Ermitteln Sie die Anzahl der Kunden, die in der Textdatei erfasst sind. (Was nutzen Sie hierbei? Eine Eigenschaft? Eine Methode? Schreiben Sie ein eigenes Skript?)

3. Öffnen Sie die Textdatei „kundenliste“ und positionieren Sie den Cursor ans Ende der letzten Zeile. Drücken Sie nun zweimal Return. Speichern und schließen Sie dann die Datei wieder. Ermitteln Sie erneut die Anzahl der Kundeneinträge. Was fällt Ihnen auf?

4. Ermitteln Sie die Häufigkeit des Namens „Meyer“ in der Datei Kundenliste. Tipp: informieren Sie sich über die Split()-Methode.

5. Ermitteln Sie die Kundennummern aller Kunden mit dem Nachnamen Meyer.

6. Erstellen Sie zu den Aufgaben 4 und 5 das zugehörige Struktogramm.

**Zusatzaufgabe:** Erstellen Sie eine Textdatei und füllen Sie diese in drei Zeilen mit beliebigem Text. Lesen Sie den Inhalt dieser Textdatei nun vollständig aus. Tipp: Informieren Sie sich hierbei über das Cmdlet Get-Content. Anschließend soll der komplette Inhalt der Textdatei in Großbuchstaben umgewandelt werden!

Überleitung zur EOF-Steuerung an dieser Stelle.

<https://www.youtube.com/watch?v=aN5IUWU6PnE> ab Minute 13.11 Output in Dateien!

### 2. Exkurs – EOF-Steuerung

#### Ausgangssituation

Anlässlich des 10-jährigen Firmenjubiläums möchte die InfoSys-GmbH die umsatzstärksten Kunden zu einer Feier einladen. Zu diesem Zweck sollen alle Kunden in die Kategorien A, B oder C eingeteilt werden. A-Kunden sind Kunden, deren laufender Umsatz größer ist als 10.000 EUR, B-Kunden liegen zwischen 1.000 EUR und 10.000 EUR und C-Kunden liegen unter 1.000 EUR.

#### Auszug aus der Kundendatei:

KNR	Name	Lfd. Umsatz in EUR
1005	Andreas Weber	2.735,50
1003	Elke Schmidt	127,50
1006	Adventos GmbH	33.900,00
1007	Franz Berger	938,00
1008	Erick Kästner	4.122,90
1001	Weller & Co KG	10.341,73
1002	Elmax KG	3.169,00
1010	Ferber KG	2.500,00

#### Arbeitsauftrag

Ihre Aufgabe ist es, einen Report zu erstellen, in dem hinter jedem Kundendatensatz die entsprechende Kategorie (A,B oder C) steht. Dazu soll ein Programm geschrieben werden, welches die Kundendatei ausgibt. Zeichnen Sie außerdem dazu das Struktogramm!

#### Struktogramm

### 3. In eine Textdatei schreiben

1. Der Kundenliste sollen fortlaufend weitere Kundendaten hinzugefügt werden. Schreiben Sie ein entsprechendes Skript! Erkunden Sie drei verschiedene Varianten und setzen Sie diese um.

--	--	--

#### Beispiel für Add-Content

```
c:\s
$path = "C:\test\addcontent.txt"
add-content $path "Zeile1"
add-content $path "Zeile2"
add-content $path "Zeile3"
```

## Auswählen eines Dateipfades mit der Klasse OpenFileDialog

<http://www.powershellpraxis.de/index.php/pfade#1.4.2%20Pfadeingabe%20mit%20Dialogfeldern>

Zeitmessung – Dauer eines Skripts/ Skriptblock , Zeitausgabe

<http://stackoverflow.com/questions/13615676/what-does-variablename-mean-in-powershell>

<http://ps.stefanrehwald.de/?p=166> Zeitmessung

<http://blog.stefanrehwald.de/2013/03/05/powershell-04-textdatei-auslesen-bearbeiten-anlegen-befullen/>

-  
-  
-

## 4. CSV-Dateien

<https://technet.microsoft.com/library/ee176874.aspx>

## 5. Binärdateien

Schwichtenberg 4.0

## 6. Batch-Dateien

**Batchdatei über die Powershell anlegen** (Weltner Workshop 3.0 S. 185 ff.)

notepad c:\test\test.bat

### Code in der Batch-Datei:

```
@echo off
Cls
Echo Programmauswahl leicht gemacht:
Echo 1: Notepad
Echo 2: Regedit
Echo 3: Explorer
Choice /n /c:123 /m "Ihre Auswahl (1, 2 oder 3)?"
if errorlevel == 3 goto three
if errorlevel == 2 goto two
if errorlevel == 1 goto one
goto end
:three
explorer.exe
goto end
:two
regedit.exe
goto end
:one
notepad.exe
:end
```

### Aufruf der Batch-Datei in der Powershell

& c:\test\test.bat

Aber warum eine Batch-Datei codieren, wenn es die Powershell gibt? Das geht auch kürzer und einfacher:

**Powershellcode:**

```
cls
"Programmauswahl leicht gemacht:
Notepad
Regedit
Explorer"

Choice /n /c:123 /m "Ihre Auswahl (1, 2 oder 3)?"

switch($LASTEXITCODE)
{
    1 {notepad}
    2 {regedit}
    3 {explorer}
}
```

Nur ausführbar in der Powershellkonsole!