



# Windows-X

Franz Fiala

Mein meistbenutzter Befehl im Windows-Tastenkürzel-Dschungel ist `Windows` (X). Bild rechts. Das sind Kurzeinstiege in die Verwaltung. Daneben aber immer noch `Ctrl` `Alt` `Del` (Strg `Alt` `Entf`), weil man damit Prozesse löschen kann und auch direkt zu den Services (Diensten) gelangt, etwas, was bei `Windows` (X) fehlt, nämlich das Menü *Administrative Tools* (Verwaltung).

Einen DOS-Prompt sucht man in diesem Menü vergeblich, dafür gibt es einen direkten Aufruf der Power-Shell, die viel mächtiger als `cmd.exe` ist.

## PowerShell

Die Skriptsprache von Windows ist heute PowerShell. Sie war es nicht immer. Anfangs gab es das rudimentäre Skripting aus der DOS-Welt. Danach kam die Zeit des Windows Scripting Host mit den zwei Sprachen Visual Basic Script und CScript.

Eine bedeutende Verallgemeinerung ist nun PowerShell geworden, eine C-ähnliche Sprache, die Zugriff auf das System erlaubt. Jemand, der mit dem DotNet-Framework vertraut ist, kann seine Kenntnisse unmittelbar zur Anwendung bringen.

Aber auch ohne diese Vorkenntnisse kann man mit den zahlreichen Beispielen seine eigene Hexenküche gestalten.

Die PowerShell ist während der letzten Jahre zu einem festen Bestandteil von Windows geworden und der prominente Platz im Systemmenü zeigt, wie unentbehrlich dieses Werkzeug geworden ist.

Die gute Nachricht: die meisten DOS-Befehle (Kommandozeilen-Befehle) funktionieren auch in der Power-Shell. Eine Übersicht gibt die Tabelle auf der folgenden Seite. Ein ‚A‘ neben dem Hackerl bedeutet, dass man administrative Rechte braucht, um den Befehl ausführen zu können.

Manche Befehle sind identisch mit den jeweiligen DOS-Befehlen, andere sind nur namensgleich und verwenden eine abweichende Syntax.

## Beispiel

`cd` oder `chdir` ist ein Alias (Synonym) zum PowerShell-Kommando `Set-Location`. Um Genaueres zu einem Befehl zu erfahren gibt man ein

```
help cd
```

Auch `help` ist ein Alias zum Power-Shell-Kommando `Get-Help`.

Um alle Kommandos zu erfahren, verwendet man `Show-Command`. Das öffnet ein eigenes Fenster mit allen 1238 Kommandos. Zum Vergleich, das `Help`-Kommando von `cmd.exe` umfasst 83 Befehle. Im An-

hang zur PDF-Version dieses Artikels ([Seite 33](#)) und bei der Web-Version findet man einen Mächtigkeitsvergleich zwischen dem Befehlsvorrat von der klassischen Windows-Konsole und PowerShell.

Wer sich jetzt mit Schaudern abwendet, weil er meint, er müsse wieder zurück in die Kommandozeilenwelt, kann ruhig weiterlesen, denn es gibt ausreichend Unterstützung in Form erklärender Programme.

Ich arbeite aber lieber mit der strukturierten Hilfe `PowerShell.chm` ([bei der Web-Version](#)).

Mit dem Betriebssystem mitgeliefert wird Windows PowerShell ISE (*Integrated Scripting Environment*). Ist nicht schlecht aber mir hat die (kostenlose) PowerGui von Quest/Dell besser gefallen. Der Grund: man kann bei einem Haltepunkt den Inhalt aller Variablen unmittelbar feststellen und die kontextsensitive Hilfe ist ausgefeilter.

## Beispielanwendung Formatierung

Ich verwende die Power-Shell für kleine Tools, die den Alltag eines Redakteurs erleichtern und stelle hier meine Arbeitsweise an Hand eines Beispiels zur Textformatierung vor ([siehe Kode auf der nächsten Seite](#)).

Ich bekomme von verschiedenen Autoren Texte. Alle diese Texte und auch die eigenen erfordern ein gewisses Maß an Handarbeit, bis sie schließlich in der schlichten Form sind, wie sie für die PCNEWS benötigt wird.

Der zu formatierende Text wird in einer Datei `s:\desktop\text.txt` gespeichert, das Ergebnis ist in der Datei `s:\desktop\text_formatiert.txt`.

```
$Text = [string]::Join("`n", (Get-Content $input_path -Encoding Default))
```

Die als ein String-Array eingelesenen Textzeilen werden mit dem `Join`-Befehl mit einem Zeilenvorschub-Zeichen zum String `$Text` verkettet.

## Formatierungs-Beispiele

- Wichtigste Formatierungen sind die Entfernung mehrfacher Zeilenumbrüche und Leerzeichen.

```
$Text = $Text -replace("[\t ]+"," ")
$Text = $Text -replace("(^n)+","n")
```

`[\t ]+` Tabulatoren oder Leerzeichen, werden, wenn sie ein oder mehrmals vorkommen (Pluszeichen) durch ein einzelnes Leerzeichen " " ersetzt.

`(^n)+` ein oder mehrere Zeilenumbrüche.

- Aber auch unbeabsichtigte Leerzeichen vor einem Punkt, Beistrich oder einer Klammer müssen entfernt werden, ebenso Tabulatoren.

- Da der Duden die Anrede in der zweiten

## Programme und Features

Energieoptionen

Ereignisanzeige

System

Geräte-Manager

Netzwerkverbindungen

Datenträgerverwaltung

Computerverwaltung

Windows PowerShell

Windows PowerShell (Administrator)

Task-Manager

Systemsteuerung

Explorer

Suchen

`Windows` (X)

Ausführen

Herunterfahren oder abmelden

Desktop

Person sowohl in Großschreibung als auch in Kleinschreibung erlaubt, hat sich der Redakteur für die erstere Version entschieden und handelte sich damit jede Menge Handarbeit ein, die jetzt eben ein Skript erledigt.

```
$Text = $Text -replace(" d(u|ir|ich|ein|eine|einer|eines|einen|einem)([ \.,,])"," D$1$2')
```

`d` Leerzeichen gefolgt von einem `d` gefolgt von einer der angegebenen Buchstabengruppen, gefolgt von

`([ \.,,])` Leerzeichen, Punkt oder Beistrich.

```
$Text = $Text -replace(" 1 "," ein ")
$Text = $Text -replace(" 2 "," zwei ")
```

Zahlen bis vier werden ausgeschrieben.

```
$Text = $Text -replace("\.\.\.\.*","...")
```

```
$Text = $Text -replace("\.\.", "... ")
```

`\.\.\.\.*` Drei oder mehr aufeinander folgende Punkte werden durch das Zeichen "..." ersetzt.

Wenn man für solche Routineaufgabe ein Skript anwenden kann, erspart man viel Arbeit und beseitigt automatisch auch den einen oder anderen Flüchtigkeitsfehler.

## Links

Die wichtigsten Regex-Regeln

<http://d.pcnews.at/pdf/n1390025.pdf>

PowerShell Kurzreferenz

<http://d.pcnews.at/pdf/n1390026.pdf>

Scripting with Windows PowerShell

<https://technet.microsoft.com/de-de/library/bb978526.aspx>

PowerShell ISE

<https://technet.microsoft.com/de-de/library/dd315244.aspx>

PowerGui

<http://en.community.dell.com/techcenter/powergui>



## #Format PCNEWS Text

```
#
cls
$input_path = 's:\desktop\text.txt'
$output_file = 's:\desktop\text_formatiert.txt'
$regex = '\b[A-Za-z0-9._%~]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}\b'
$Text = [string]::Join("`n", (Get-Content $input_path -Encoding Default))
$Text = $Text -replace("[\t ]+", " ")
$Text = $Text -replace("(`n)+", "`n")
$Text = $Text -replace("`n\s+`n", "`n")
$Text = $Text -replace("^ ", "")
$Text = $Text -replace(" +\)", ")")
$Text = $Text -replace("\( +", "(")
$Text = $Text -replace(" +\.", ".")
$Text = $Text -replace(" +\,", ",")
$Text = $Text -replace(" +\?", "?")
$Text = $Text -replace(" dir ", " Dir ")
$Text = $Text
    -replace(" d(u|ir|ich|ein|eine|einer|eines|einen|einem)([ \.\,])", ' D$1$2')
$Text = $Text -replace("iess", "ieß")
$Text = $Text -replace("öss", "öß")
$Text = $Text -replace(" weiss ", " weiß ")
$Text = $Text -replace(" t(ools?)([ \.\,])", ' T$1$2')
$Text = $Text -replace(" s(creenshots?)([ \.\,])", ' S$1$2')
$Text = $Text -replace(" f(iles?)([ \.\,])", ' F$1$2')
$Text = $Text -replace(" mb ", ' MB ')
$Text = $Text -replace(" Mbyte ", ' MByte ')
$Text = $Text -replace(" daß ", " dass ")
$Text = $Text -replace(" gross ", " groß ")
$Text = $Text -replace(" grosse ", " große ")
$Text = $Text -replace(" weiss", " weiß")
$Text = $Text -replace(" Gruss ", " Gruß ")
$Text = $Text -replace(" 1 ", " ein ")
$Text = $Text -replace(" 2 ", " zwei ")
$Text = $Text -replace(" 3 ", " drei ")
$Text = $Text -replace(" 4 ", " vier ")
$Text = $Text -replace(" 5 ", " fünf ")
$Text = $Text -replace("\.\.\.\.\.*", "... ")
$Text = $Text -replace("\.\.", "... ")
$Text = $Text -replace("[\t ]+", " ")
$Text = $Text -split "`n"
#for ($i=0; $i -lt $Text.Count; $i++) {
# $Text[$i] = $Text[$i] + "`n"
#}
$Text
$Text > $output_file
```

