

# Batch-Programmierung/ Druckversion

---

< [Batch-Programmierung](#)

## Batch-Programmierung

---

### Wikibooks

---

## Allgemeine Information

### Inhaltsverzeichnis

---

#### Allgemeine Information

In diesem Buch behandelte (und nicht behandelte) Skriptsprachen

Was sind und wie funktionieren Batchprogramme?

Nutzung von Batchprogrammen

Wie erstelle ich eine Batchdatei?

Weiterführende Informationen und Links

#### Wichtige DOS-Kommandos

##### Befehle

assoc

attrib

Mögliche Eigenschaften

Beispiele

cacls

Beispiele

Rechte vergeben

Rechte entziehen

cd / chdir

comp oder fc

(x)copy/robocopy

del / erase

deltree

dir

## Kombinationen

edit

find

format

loadhigh

md / mkdir

more

move

path

rd / rmdir

ren / rename

start

taskkill

tasklist

type

Anwendungsbeispiel(e)

Pfad der laufenden Batchdatei

Fußnoten

## **Batch Befehle**

Wichtiger Hinweis zu diesem Abschnitt

Vorsicht Leerzeichen!

@

! (Ausrufezeichen)

:(Doppelpunkt)

:: (doppelter Doppelpunkt)

CALL

CHCP

CLS

CMD

COLOR

COMMAND

DATE

ECHO

Benutzereingaben mit ECHO

ERRORLEVEL

ENDLOCAL / (SETLOCAL)

EXIT

FOR

GOTO

IF

Syntaxvergleiche

Hinweis zu UND- bzw. ODER-Verknüpfung

KEYB

PAUSE

PROMPT

PUSHD / POPD

REM

[RENAME](#)[SET](#)[SETLOCAL / \(ENDLOCAL\)](#)[TIME](#)[TITLE](#)[>, >>](#)[TIMEOUT](#)

### **Batch Operatoren**

[&](#)[|](#)[&&](#)[||](#)[<](#)[>](#)[>>](#)

### **Erweiterungen unter Windows NT**

[Variablen Kurzübersicht](#)[Scriptvariablen und Systemvariablen](#)[Benutzereingaben in Variablen speichern](#)[Zeichenketten in Variablen manipulieren \(Stringoperationen\)](#)[Mit Variablen rechnen \(Arithmetische Operationen\)](#)[Variablenerstellung, -zuweisung und -abruf](#)[Gültigkeitsbereich von Variablen](#)[Allgemeine Info zum Gültigkeitsbereich von Variablen](#)[Gültigkeitsbereich setzen \(setlocal\)](#)[Umgebungsvariablen \(Windows Environment Variables\)](#)[Weitere "Variablenarten"](#)[Übergabeparameter](#)[Parametervariablen in For-Schleifen](#)[Stringoperationen \(Manipulation von Strings in Variablen\)](#)[Arithmetische Operationen \(set /a\)](#)[Ein- und Ausgaben in Variablen speichern \("set /p"\)](#)[Benutzereingaben](#)[Umgeleitete Ausgaben](#)[Sonderzeichen verwenden](#)

### **Programmierungshilfen**

[Ändern des Editors zum Bearbeiten von Batchdateien](#)[Datum und Uhrzeit anzeigen](#)[Lokalisierungsunabhängige Datumsermittlung](#)[Ausgaben besser anzeigen](#)[Unterroutinen und Unterprogramme](#)[Benutzereingaben mittels "set /P"](#)[stdout in Umgebungsvariable speichern](#)[Beispiel:](#)[Dateien und Verzeichnisse auflisten](#)[Pause](#)[Minimiert ausführen](#)

Mittels start /LOW die Priorität festlegen

Probleme mit Variablen

Das Problem

Lösung: cmd.exe /V:ON

Lösung: setlocal EnableDelayedExpansion

Ausgaben/Fehler unterdrücken

Professionelle Message-Fenster erzeugen

Falls das nicht funktioniert

Status über bearbeitete Zeilen ausgeben

Einzelnachweise

## **Netzwerkumgebung**

ping

ipconfig

ipconfig /all

ipconfig /renew

ipconfig /flushdns

ipconfig /release

tracert

netstat

netsh

NET

NET USE - Netzlaufwerke verbinden

NET SEND - Nachrichten an andere Rechner senden

NET START

NET STOP

NET VIEW

NET USER

## **Beispiele**

Regedit.exe

TaskList-Abfrage

Dienste einrichten

Windows XP SP2: installierte Patches/Updates auflisten

Printdateien direkt an den Drucker senden

Copy

Message-Ping

Defrag mit Endlos-Schleife

Dateiinformationen anzeigen

Alle verfügbaren PCs im aktuellen Netzwerk suchen

Hosts Datei mit einem Aliasnamen und der aktuellen IP-Adresse aktualisieren

Windowsversion herausfinden

TEMP-Verzeichnis löschen

FLV/MP4/WAV zu MP3 konvertieren (FFMPEG + Batch)

## **Zusatz-Tools**

### **Referenz**

help

help als Einzelbefehl

help in Kombination mit anderen Befehlen

## WebLinks

[Foren / Hilfe zur Batch-Programmierung finden](#)

[Ungeordnete Links \(Seiten mit tutorien, Referenzen, Beispielen etc.\)](#)

[Tutorials](#)

[Bücher](#)

[Referenzen](#)

## In diesem Buch behandelte (und nicht behandelte) Skriptsprachen

---

In diesem Buch geht es primär um das Erstellen von Batchdateien/Skripten für den Microsoft Windows Kommandozeileninterpreter ([cmd.exe](#) für Windows XP / Vista / 7). Informationen zu Batchdateien unter DOS (und auf DOS basierenden Windowsversionen wie z.B. Windows 98) werden teilweise miterläutert. Unterschiede zur Verwendung der Befehle in den unterschiedlichen Windowsversionen werden unter der jeweiligen Befehlsbeschreibung erläutert.

Andere Betriebssysteme bieten vergleichbare und teilweise wesentlich mächtigere Funktionen (wie z. B. die in der Linux-Welt weit verbreitete Bash oder AppleScript für den Mac). Auch für Windows gibt es Möglichkeiten, mächtigere Skripte zu erstellen, hierzu kann der [Windows Script Host](#) oder das [Windows PowerShell](#) verwendet werden.

## Was sind und wie funktionieren Batchprogramme?

---

Batchdateien oder Batchprogramme (häufig mit Stapelverarbeitungsprogramme oder kurz Stapelprogramm übersetzt) sind meist kurze Dateien, die Befehle der Kommandozeile der Reihe nach abarbeiten. Batchprogrammierung erfüllte die grundlegenden Anforderungen an eine Programmiersprache wie z.B. `if`, `if not` und `while`. Da Batch sich in den Bereich Shells scripting einordnen lässt, kann man von einer *interpreterbasierten Skriptsprache* reden. Das heißt, dass die im Editor erstellte Textdatei nicht mit einem [Compiler](#) einmal in Maschinensprache übersetzt und in einer ausführbaren Binärdatei im `*.exe` Format gespeichert, sondern bei jedem Aufruf durch einen Interpreter zur Laufzeit in ein für den Computer verständliches Format übersetzt wird. Ein Texteditor wie MS Notepad, den Windows von Haus aus mitbringt, reicht vollkommen aus, um Batchprogramme zu schreiben. Batchprogrammierung ist speziell für die Steuerung des Betriebssystems gedacht, für die Entwicklung von Anwendungen ist sie nicht geeignet. Hierzu verwendet man Sprachen wie C/C++ und Java (nicht zu verwechseln mit JavaScript!!).

## Nutzung von Batchprogrammen

---

Viele wiederkehrende Installations- und Verwaltungsaufgaben, die man traditionell mit Tastatur und Maus ausführt, lassen sich mit Stapeldateien ausführen. Hat man genügend Kommentarzeilen in die Batchdatei eingefügt, genügt der Ausdruck der Stapeldatei meist als Dokumentation.

Kommandozeilenbefehle und Batchdateien sind keine veraltete Technologie. Auf Wunsch vieler Systemadministratoren hat Microsoft dafür gesorgt, dass Windows Server 2003 vollständig von der Kommandozeile installiert und administriert werden kann, ohne die Maus zu benutzen.

Die Begriffe "Stapelprogramm" und "Stapelverarbeitung" rühren aus der Zeit her, als Permanentpeicher auf Magnetplatten sehr teuer war. Zu dieser Zeit begab man sich mit einem Stapel Lochkarten aus Pappe, auf denen sich das Programm und die Daten in gestanzter Form befanden, zum Rechenzentrum und übergab dem Operator seinen Stapel. Die Stapel aller Benutzer wurden gesammelt und dann sequentiell an den Rechner übergeben. Nach einiger Zeit konnte man sich dann die (meist ausgedruckten) Ergebnisse abholen.

## Wie erstelle ich eine Batchdatei?

---

Wie bereits gesagt, reicht ein einfacher Editor aus. Empfehlen kann man guten Gewissens den quelloffenen und kostenlosen Notepad++ (<http://notepad-plus.sourceforge.net/de/site.htm>)-Editor, als bessere Alternative zu MS Notepad.

Unter DOS bis Windows 98:

Speichern Sie dann die Datei mit der Endung \*.bat ab. Anschließend müssen Sie nur die Datei starten: Dies können Sie entweder direkt in Windows oder indem Sie in der Eingabeaufforderung in das entsprechende Verzeichnis wechseln und den Dateinamen eingeben.

Ab Windows NT / 2000 / XP / Vista / 7:

Für Windows NT-kompatible Betriebssysteme gibt es seit Windows 2000 auch Batchdateien mit der Endung \*.cmd. Diese werden genau wie Batchdateien mit der Endung \*.bat verarbeitet bzw. ausgeführt. Nach allgemeiner Konvention sollte man bei Batchdateien erstellt für Windows NT die Endung "CMD" wählen. Die Endung \*.cmd wurde ursprünglich aus Kompatibilitätsgründen zu OS/2, einem Betriebssystem, das MS ursprünglich in Kooperation mit IBM entwickelte, mit Windows NT 3.x eingeführt.

## Weiterführende Informationen und Links

---

- Allgemeine Informationen: Wikipedia: batch
- Infos zur cmd.exe: Wikipedia: cmd.exe
- Die für **Windows XP** verfügbaren Kommandos können auch im Hilfesystem von Windows direkt eingesehen werden:
  - START - Hilfe und Support
  - "Befehlszeilenreferenz A-Z" in das Suchfeld eintragen. (Für englisches Windows "Command-line reference A-Z")

## Wichtige DOS-Kommandos

Bei den auch sogenannten *Batch*-Befehlen wird grundsätzlich zwischen *internen* und *externen* Befehlen unterscheiden (vergleiche FreeDOS-Kompendium: Befehle). Die Enthaltenen (*Internen*) sind Bestandteil des jeweiligen Befehlsinterpreters selbst, während die nicht enthaltenen (*externen*) Befehle gegebenenfalls als eigene Programme vorliegen. Unabhängig davon sind **Befehle** (oder auch sogenannte *Kommandos*), **die wichtig für die Batch-Programmierung sein könnten**, im Folgenden aufgeführt.

# Befehle

---

Einzelne Befehle:

## assoc

Mit Hilfe dieses Befehls kann man die Dateierweiterungen bearbeiten bzw. anzeigen lassen.

```
assoc [.dateiendung[=Dateibeschreibung]]
```

## attrib

Ändert die Eigenschaften (oder auch sogenannten *Attribute*) unter anderem von Dateien und Ordnern.

### Mögliche Eigenschaften

Folgende (aus *DOS-FAT* geerbte) Eigenschaften sind möglich:

- R – für „schreibgeschützt“ (im Englischen ‚*read only*‘ genannt)
- H – ... „versteckt“ (englisch *hidden*)
- S – ... „systemversteckt“ oder auch „zum Betriebssystem gehörend“ (mit gesetzter Eigenschaft auch „Systemdatei“ genannt)
- D – ... „Ordner“ oder auch „Verzeichnis“ (englisch *directory*); diese Eigenschaft war bis einschließlich *Windows XP* nicht sichtbar und wurde frühestens mit *Windows Vista* auch im *Windows-Explorer* anzeigbar
- A – ... „zu archivierende Datei“ (auch „Archiv“ genannt)

Mit *Windows NT* (und dessen *NTFS*) wurden unter anderem folgende Eigenschaften ergänzt:<sup>[1]</sup>

- N – für „keine“ (englisch *none*)
- L – ... „(symbolischen Datei-)Verknüpfung“ oder „(Ordner-)Verbindung“<sup>[2]</sup> (auch *Abzweigungspunkt* und im Englischen allgemein *link* genannt)
- C – ... „verdichtet“, „zusammengepreßt“ oder auch „komprimiert“ (englisch *compressed*)
- I – ... „nicht verschlagwortet“ oder auch „nicht indiziert“<sup>[3]</sup>
- E – ... „verschlüsselt“ (englisch *encrypted*); ist (bisher) ggf. nur im *Windows-Explorer* anzeig- und änderbar

Weitere Eigenschaften:

- O – für das (im Amerikanisch-Englischen) sogenannte ‚*Offline attribute*‘<sup>[4]</sup>
- X – ... ‚*No scrub file attribute*‘,<sup>[4]</sup> (wahrscheinlich) für „nicht löschar“ oder „nicht löschen“
- V – ... ‚*Integrity attribute*‘,<sup>[4]</sup> wahrscheinlich für *Chkdsk* (und ähnliche Hilfsanwendungen zur Integritätsprüfung)
- P – ... ‚*Pinned attribute*‘<sup>[4]</sup>
- U – ... ‚*Unpinned attribute*‘<sup>[4]</sup>

## Beispiele

In folgendem Beispiel wird einer zuvor angelegten und „Beispieldatei.txt“ genannten Daten der (ggf. vorhandene weiche<sup>[5]</sup>) Schreibschutz entzogen und zudem die selbe Datei versteckt:

```
attrib Beispieldatei.txt -r +h
```

Im Folgendem wird einem zuvor angelegten und „Beispielordner“ genanntem Ordner der (weiche) Schreibschutz hinzugefügt:

```
attrib Beispielordner +r
```

Und im Folgendem wird einem zuvor ebenso angelegten Ordner (mit +i) das auch sogenannte *Index*-Bit angefügt, welches bewirkt, daß dessen Inhalte nicht mehr (wie in der Vorgabe von *Microsoft*, ohne I) für eine (angeblich) schnelle Dateisuche *indiziert* (oder übersetzt: *verschlagwortet*) werden. Zudem sind dort am Zeilenende zwei Schalter angefügt, um (mit /s) neben dem genannten Ordner auch all seine Unterordner und (mit /d) auch jene Ordner selbst entsprechend von der (üblicherweise im Hintergrund – über den in *Windows XP* auch sogenannten „*Indexdienst*“ – ausgeführten) Verschlagwortung auszuschließen:

```
attrib Beispielordner +i /s /d
```

## cacls

Ändert die Zugriffsrechte (über die im Amerikanisch-Englischen sogenannten *Access Control Lists*) auf NTFS-Partitionen oder zeigt diese an.

## Beispiele

### Rechte vergeben

Der folgende Befehl fügt beispielsweise dem Benutzer „Albert Einstein“ (mit der Eigenschaft C, wie *change*) das Recht zum „Ändern“ hinzu – dies (über die Platzhalter \*.\* ) für alle Dateien und Ordner im aktuellen und (mit dem Schalter /T, wie *tree*) zudem auch in allen Unterordnern (oder auch im dazu untergeordneten Verzeichnisbaum).

```
cacls *.* /E /T /G "Albert Einstein:C"
```

Über den Schalter /E werden zudem die Rechte bearbeitet, anstatt sie (wie in der Vorgabe – ohne den Schalter) zu ersetzen.

### Rechte entziehen

Der folgende Befehl hebt die zuvor vergebene Berechtigung wieder auf.

```
cacls *.* /E /T /R "Albert Einstein"
```

Über den folgenden Befehl können die Zugriffsrechte für die einzelne ebenda genannte Datei so geändert (genauer über den Schalter /P so ersetzt) werden, daß jedem (englisch *everyone*) alle Rechte (mit der Eigenschaft N, wie das englisch *none*, für „keine [Rechte]“) entzogen werden.<sup>[6]</sup>

```
cacls %windir%\system32\jscript.dll /E /P everyone:N
```

## cd / chdir

Über den `cd`- oder auch `chdir`-Befehl (kurz für das englische *change directory*, wörtlich „wechsel Verzeichnis“) kann das aktuelle Verzeichnis gewechselt werden. Beachte: Nach dem `cd` folgt üblicherweise ein Leerzeichen, welches bei nachfolgendem `..` oder `\` (je nach Betriebssystemversion) aber auch weggelassen werden kann. Wird der Befehl ohne Übergabewert ausgeführt, so wird der Pfad des aktuellen Verzeichnisses ausgegeben.

relative Angaben:

Die Eingabe von "`cd ..`" wechselt in das nächsthöhere Verzeichnis ("`cd ..\..`" zwei Ebenen höher usw.).

Die Eingabe von "`cd ORDNERNAME`" wechselt in den Unterordner `ORDNERNAME`.

absolute Angaben:

Die Eingabe von "`cd %~dpo`" wechselt in das Verzeichnis aus dem die Batchdatei aufgerufen wurde.

Die Eingabe von "`cd \`" wechselt in das Hauptverzeichnis auf dem derzeitig befindlichen Laufwerk.

Die Eingabe von "`cd \ORDNERNAME`" wechselt absolut in den Ordner `ORDNERNAME` auf dem derzeitig befindlichen Laufwerk.

Die Eingabe von "`LAUFWERKSBUCHSTABE:`" (ohne vorangestelltes "`cd`") wechselt in das aktuelle Verzeichnis des gewählten Laufwerks `LAUFWERKSBUCHSTABE`.

Die Eingabe von "`cd LAUFWERKSBUCHSTABE:\ORDNERNAME`" wechselt absolut in den Ordner `ORDNERNAME` auf dem gewählten Laufwerk von `LAUFWERKSBUCHSTABE` jedoch nicht von einem Laufwerk auf ein anderes.

Pfadangaben mit Leerschlägen (Bsp "`C:\program files\`") müssen in Anführungs-/Schlusszeichen eingeschlossen sein.

Der Parameter `/D` bewirkt, dass zusätzlich zum Wechseln des Verzeichnisses auch das aktuelle Laufwerk gewechselt wird. Der Befehl "`cd /D %~dpo`" wechselt in den Ordner des Batch-Skripts. (sinnvoll bei Windows-Doppelklicks)

Wildcards:

Ist die Pfadangabe eindeutig kann der Ordnername durch Nutzen von Wildcards (z.b. `\win*` für `\windows` oder `\winnt`) abgekürzt werden. Dies ist jedoch im Batchbetrieb nicht ratsam.

Remotesysteme:

Pfade von nicht als Laufwerken verbundenen Remotesystemen können (Berechtigung vorausgesetzt) entweder über die Administrative Freigabe oder durch das Verbinden eines Netzlaufwerks angesprochen werden. Beispiel für administrative Freigabe: `\\[computername]\[laufwerk]$\[Weiterer Pfad]`

Pfad in Variable:

Der Befehl `set PFAD=%cd%` schreibt das aktuelle Verzeichnis in die Variable `%PFAD%`. (Als letztes im Batch-Script mit `cd %PFAD%` wieder ins Startverzeichnis wechseln!)

## comp oder fc

Vergleicht den Inhalt zweier Dateien und stellt die Unterschiede dar (**compare** / **file compare**).

## (x)copy/robocopy

Mit dem Befehl `copy` können eine Datei oder mehrere Dateien kopiert werden.

`xcopy` ist die erweiterte Version von `copy`, die mehr Optionen bietet. Diese ist nicht in allen Windows-Versionen verfügbar.

Das weitaus mächtigere `robocopy` ist ab Vista standardmäßig wieder verfügbar, bei Windows XP und 2000 fehlt dieses Programm zunächst. Microsoft stellt `robocopy` aber im Windows Server 2003 Resource Kit Tools 2003 auch für diese Betriebssystemversionen bereit.

Mit dem Befehl `copy /b "DATEI" + "DATEI" "NEUE DATEI"` kann man Dateien zusammenfügen.

## del / erase

Löscht eine Datei, die sich im aktuellen Verzeichnis befindet. Als Parameter muss mindestens der Dateiname angegeben werden. Weitere Parameter sind optional:

- `/p` – sollen mehrere Dateien gelöscht werden, muss jede einzelne Löschung bestätigt werden.
- `/f` – auch schreibgeschützte Dateien werden gelöscht.
- `/s` – löscht auch Dateien, die sich in den Unterverzeichnissen befinden.
- `/q` – Unterbindet Bestätigung.

## Beispiel:

```
erase "H:\zulöschende Datei.bat"
```

## deltree

Löscht Verzeichnis(se) inkl. untergeordnete Dateien.

Unter Windows XP lautet der Befehl **rmdir**. Deltree kennt es nicht mehr. Anmerkung: `rmdir/rd` versteht keine Wildcards ('\*' oder '?') im Verzeichnisnamen.

## dir

Der Befehl `dir` (kurz für das englische *directory*) zeigt alle Verzeichnisse und Unterverzeichnisse an. Verzeichnisse werden seit Windows 95 als Ordner bezeichnet, unterscheiden sich allerdings nicht von Verzeichnissen.

Der `dir`-Befehl hat zahlreiche Optionen:

- `/b` – Nur der Dateiname wird ausgegeben. Die Ausgabe aller weiteren Informationen wie beispielsweise Datum, Größe oder Datenträgerbezeichnung wird unterdrückt.
- `/c` – Die Dateigröße wird mit einem Punkt als Trennzeichen zwischen jeweils drei Dezimalstellen angegeben, also beispielsweise 343.232 (Standardeinstellung). Soll dies unterdrückt werden, geben Sie den Schalter `/-c` an.
- `/p` – Bei jedem Seitenumbruch wird die Ausgabe unterbrochen, bis der Anwender eine Taste drückt.
- `/on` – Gibt das Verzeichnis alphabetisch sortiert aus.
- `/w` – Mehrere Einträge in einer Zeile werden angezeigt.
- `/ah` – Versteckte Dateien werden angezeigt.
- `/a` – Alle versteckten- und System-Dateien werden angezeigt.

- /s – Zeigt auch den Inhalt der Unterverzeichnisse an.

Mit dem Befehl `dir [Dateiname]` ist es auch möglich, Dateien im aktuellen Verzeichnis zu suchen. Alternativ kann auch der Platzhalter `*` (bspw.: `dir *.*`) verwendet werden, um nach bestimmten Dateien zu suchen.

Mit `dir *` werden nur Objekte ohne Dateinamen-Erweiterung angezeigt. In der Regel haben Ordner keine Erweiterung und Dateien eine Erweiterung. In diesem Fall werden nur Ordner angezeigt. Gewöhnliche Ordner werden mit `<DIR>` vor dessen Namen gekennzeichnet und Ordnerverbindungen mit `<VERBINDUNG>` (oder auf nicht oder nur teilweise übersetzten Cmd-Ausgaben mit `<JUNCTION>`).

## Kombinationen

```
dir /w/o/s/p    durchsucht den aktuellen Pfad inklusive Unterordner
nach einer Datei und stellt das Ergebnis seitenweise dar!
```

## edit

Mit `edit` wird ein Editor geöffnet, zum Beispiel zum Bearbeiten von Batch- oder Textdateien.

Ab Windows NT nur noch in 32 bit Systemen enthalten.

## find

Mit dem Befehl `find` kann - auch in mehreren Dateien - nach einer Zeichenfolge gesucht werden.

Auch `find` kann mit Parametern gesteuert werden:

- `find /i` ignoriert Groß-/Kleinschreibung beim Suchbegriff
- `find /v` Zeigt alle Zeilen an, die die Zeichenfolge NICHT enthalten.
- `find /c` Zeigt nur die Anzahl der die Zeichenfolge enthaltenden Zeilen an.
- `find /n` Zeigt die Zeilen mit ihren Zeilennummern an.

## format

Formatiert die Festplatte oder Diskette. Mit `format c:` wird die Festplatte `c:` formatiert. Mit `format a:` wird die Diskette im Laufwerk `a:` formatiert. Mit `format c: -s` bzw. `format a: -s` kann man die Festplatte/Diskette mit Systemdateien formatieren, so dass sie nachher bootfähig sind.

## loadhigh

Lädt Programme in freie UMB Upper-Memory-Blocks

## md / mkdir

Der Befehl `md` (**m**ake **d**irectory) legt ein neues Verzeichnis an.

## more

Mit der Ausgabeumleitung | `more (bspw.: type [Dateiname] | more )` wird bei Seitenwechsel zur Bestätigung aufgefordert.

Damit besteht die Möglichkeit über mehrere Seiten führende Dokumente anzuzeigen.

`more` ist der einzige Befehl, bei der auch in der DOS-Welt die sogenannte Pipe verwendet wird. Bei der Pipe handelt es sich um das Zeichen | (fehlt es auf der Tastatur, ist es auch bei hierzu gedrückt gehaltener ALT-Taste und der Eingabe der Zahlenfolge 124 auf dem Ziffernblock zu erzeugen; das Zeichen erscheint dann nach dem Loslassen der ALT-Taste). Mit diesem Zeichen ist es möglich, zwei Befehle miteinander auf eine spezielle Weise zu verbinden. Dabei wird der erste Befehl ausgeführt, seine Ausgaben werden jedoch nicht auf den Bildschirm geschrieben, sondern in einen Puffer. Der zweite Befehl wird auf eine Weise gestartet, dass er seine Eingaben nicht von der Tastatur, sondern von ebendiesem Puffer bekommt.

Mit `more` ist es ebenfalls möglich den Inhalt einer Datei an eine andere Datei anzuhängen.

So kann man zum Beispiel mit: `more "C:\eins.txt" >> "D:\zwei.txt"` den Inhalt der "eins.txt" an den Inhalt der "zwei.txt" anhängen. "eins.txt" wird dabei nicht verändert.

## move

Verschiebt (englisch *moves*) eine oder mehrere Dateien von einem Verzeichnis in ein anderes. Dieser Befehl kann auch zu Umbenennung (englisch rename) mißbraucht werden.

### Beispiel:

```
move .\Beispieldatei.txt ".\Unterordner\Beispieldatei.txt"
```

## path

Durch die Angabe eines Pfades wird das Betriebssystem auf der Kommandozeile dazu aufgefordert, neben dem aktuellen Verzeichnis auch in weiteren Verzeichnissen zu suchen. Wird der Befehl ohne Parameter ausgeführt, so werden alle aktuellen Pfade ausgegeben.

## rd / rmdir

Der Befehl `rd` oder `rmdir` (remove directory) löscht ein Verzeichnis, wenn dieses leer ist.

Parameter:

- `/s` Löscht einen ganzen Verzeichnisbaum mit Dateien
- `/q` Löscht ohne Nachfrage ganze Verzeichnisbäume. Kann in Batch-Dateien praktisch sein.

Der Verzeichnisname darf keine Platzhalter (wie `*` oder `?`) enthalten.

### Beispiel:

```
rmdir /s /q C:\temp
```

## ren / rename

Datei(en) und Ordner umbenennen – über den englischsprachigen Befehl `rename` und zudem kurz `ren`.

### Beispiel:

```
rename Beispieldatei.txt "Beispieldatei mit neuem Bezeichner.txt"
```

### start

**Syntax:** `START ["Titel"] [/D <Pfad>] [/I] [/MIN] [/MAX] [/SEPARATE | /SHARED] [/LOW | /NORMAL | /HIGH | /REALTIME] [/WAIT] [/B] [Befehl/Programm] [Parameter]`

### Optionen:

- "Titel" Der Titel des neuen Fensters.
- /D <Pfad> Startverzeichnis
- /I Die neue Umgebung soll die dem CMD.EXE beim Aufruf übergebene sein und nicht die aktuelle Umgebung.
- /MIN Startet das Fenster minimiert.
- /MAX Startet das Fenster maximiert.
- /SEPARATE Startet 16-Bit-Windows-Programm in separatem Speicherbereich.
- /SHARED Startet 16-Bit-Windows-Programm in gemeinsamen Speicherbereich.
- /LOW Startet Anwendung in IDLE-Prioritätsklasse.
- /NORMAL Startet Anwendung in der NORMAL-Prioritätsklasse.
- /HIGH Startet Anwendung in der HIGH-Prioritätsklasse.
- /REALTIME Startet Anwendung in der REALTIME-Prioritätsklasse.
- /WAIT Startet die Anwendung und wartet auf das Ende.
- /B Startet die Anwendung ohne ein neues Fenster zu öffnen. Die Anwendung ignoriert STRG+C. Wenn die Anwendung nicht selbständig STRG+C überprüft, ist STRG+UNTBR die einzige Möglichkeit, um die Anwendung abzubrechen.
- /? Gibt die Hilfe aus.

### Erklärung:

- Startet ein Programm (als neues eigenes Fenster).

### Beispiel:

Auf Windows XP: Zum Öffnen eines neuen Konsolenfensters in dem eine Datei aufgerufen wird.

```
start "Name des neuen Fensters" /WAIT /D C:\temp /NORMAL call test.bat
```

/WAIT = Wartet bis Anwendung geschlossen wird.

/NORMAL = Startet Anwendung in der NORMAL-Prioritätsklasse.

/D = Danach folgt das Startverzeichnis

**Nützliche (System)-Programme:** Hier eine Liste der nützlichen (System)-Programme, die man so ausführen kann:

regedit.exe = neuer Registrierungseditor  
(regedt32.exe = älterer Registrierungseditor)

explorer.exe = Windows Ordner Explorer  
taskmgr.exe = Windows Taskmanager  
taskeng.exe = Aufgabenplanungsmodul  
calc.exe = Taschenrechner  
mshta.exe = Scripthost für HTA (HTML) Scripting  
msconfig.exe = Autostart verzeichnis

W/CScript.exe = Beides Bestandteil des Windows Scripthostes für JS und VBS

iexplore.exe = Microsoft Windows Internet Explorer, Standardbrowser für Windows  
firefox.exe = Mozilla Firefox, Internetbrowser  
dialer.exe = Windows-Hilfeprogramm für DFÜ Einwahlverbindungen  
Notepad.exe = Standard-Textbearbeitungsprogramm von Microsoft

cmd.exe = Microsoft Windows Befehlsprozessor für Batch  
winword = Microsoft Word

## taskkill

Ab Windows XP kann man mit diesem Befehl einen Prozess beenden, beispielsweise wenn dieser nicht mehr reagiert. Unter Einbindung von /t werden auch alle untergeordneten Prozesse beendet, und wenn man /f hinzufügt, wird das Beenden erzwungen.

```
taskkill /im iexplore.exe /t /f
```

## tasklist

Dieser Befehl listet alle aktiven Prozesse auf, die gerade auf dem lokalen Rechner laufen. Damit lässt sich zum Beispiel der Name eines zu beendenden Prozesses ermitteln (siehe auch Befehl: [taskkill](#)). Mit dem hinzugefügten Schalter /nh werden die Spaltenüberschriften ausgeblendet.

Der Befehl `tasklist` ist erst ab Windows XP verfügbar.

## type

Gibt den (reinen Text-)Inhalt einer übergebenen menschenlesbaren Datei (beispielsweise im \*.bat, \*.cmd oder \*.txt-Format) aus.

```
type Beispieldatei.txt
```

# Anwendungsbeispiel(e)

---

## Pfad der laufenden Batchdatei

Es gibt Situationen, in denen es interessiert, in welchem Verzeichnispfad die laufende Batchdatei liegt; mit der Variable %~dp0 kann dieser (einschließlich dem Laufwerksbuchstaben) wie folgt (bspw. aus einer `Beispiel.bat` heraus) ausgegeben werden:

```
@echo off
echo.Dieses Script liegt im Verzeichnispfad %~dp0
pause
```

## Fußnoten

---

1. ... diese sind ggf. (auch – ähnlich wie die Eigenschaft D) erst ab *Windows Vista*, spätestens aber mit *Windows 10*, vollständig über den `attrib`-Befehl (einschließlich dessen eigener Hilfe, über `attrib /?`) anzeig- und änderbar; im *Windows-Explorer* sind diese Eigenschaften ebenso spätestens ab *Windows 10* auch (offen, in dessen Hauptfensterbereich, über die in der Vorgabe ausgeblendete Spalte „Attribute“) anzeig- und (größtenteils, wie im Folgenden beschrieben) änderbar; ... ebenda in den jeweiligen (Datei- und Ordner-)Eigenschaften, über die ggf. vorhandene Schaltfläche `E`erweitert... und dann im damit zu öffnenden Fensterchen mit der Überschrift „Erweiterte Attribute“, über die jeweiligen Einstellungen, welche (durch ein [nicht] gesetztes Häkchen) an- oder abwählbar sind und ggf. zur abschließenden Änderung durch Drücken der dortigen `O`K -Schaltfläche bestätigt werden müssen
2. in der (deutschsprachigen) Hilfe zum `mklink`-Befehl „Verzeichnisverbindung“ und in der (deutschsprachigen) Ausgabe vom `dir`-Befehl zudem kurz `<VERBINDUNG>` genannt
3. ... in der Hilfe des verwandten `dir`-Befehls (auf *Windows 10* – oder früher, frühestens wohl ab *Vista* – aufrufbar mit `dir /?`) wörtlich mit „I Nicht indizierte Dateien“ beschrieben; vergleiche zudem ggf. auch *Wiktionary:de:indizieren* und (als zugehörige Nebenform) *...:indexieren* – letzterer (Wörterbuch-)Eintrag u.a. auch mit „verschlagworten“; diese (*Index*-)Eigenschaft ist wohl auch frühestens ab *Windows Vista* über den erst dort entsprechend erweiterten `attrib`-Befehl anzeig- und änderbar
4. aus der zugehörigen (englischsprachigen) `attrib`-Hilfe (auf einem eigentlich deutschsprachigen *Windows 10*)
5. ... als Abgrenzung zum harten Schreibschutz, über die Zugriffsrechte (welche auch über den `cacls`-Befehl geändert werden können)
6. *Windows: Microsoft bringt Patch gegen Lücke im Internet Explorer* (<https://www.golem.de/news/windows-microsoft-bringt-patch-gegen-luecke-im-internet-explorer-1812-138344.html>) – *Golem*, am 20.12.2018

## Batch Befehle

### Wichtiger Hinweis zu diesem Abschnitt

---

Diese Liste ist nicht vollständig, weitere Befehle werden in den anderen Kapiteln dieses Buches erklärt. Mit der Zeit sollen alle Befehle in diese Liste übertragen werden, um aus den anderen Kapitel referenziert werden zu können.

### Vorsicht Leerzeichen!

---

Fehlplatzierte oder fehlende Leerzeichen können beim Programmieren einer Batch-Datei zu Fehlern führen. Bei den nachfolgenden Beispielen ist also auf die Platzierung von Leerzeichen und auf entsprechende Bemerkungen genau zu achten. Scheinbar grundlose Abbrüche beim Ausführen einer Batch-Datei können ebenfalls fehlplatzierten oder fehlenden Leerzeichen geschuldet sein.

@

---

Schaltet die Ausgabe der Befehlszeile auf dem Bildschirm nur für den aktuellen Befehl aus und ist selbst kein eigener Befehl.

## Syntax

@befehl

Stapelanweisung: Ab MS-DOS bis Windows 7

## Beispiel:

Inhalt

```
echo Diese Zeile wird mit Befehlszeile ausgeführt...
@echo und diese ohne!
```

Ausgabe

```
C:\>echo Diese Zeile wird mit Befehlszeile ausgeführt...
Diese Zeile wird mit Befehlszeile ausgeführt...

und diese ohne!
```

In Batch Files verhindert "@echo off" zu Beginn des Skriptes die Ausgabe aller (!) Befehlszeilen auf dem Bildschirm bis die Stapelverarbeitung beendet wird, abbricht oder mittendrin ein "@echo on" Befehl erfolgt, um z. B. Befehlszeilen tatsächlich anzuzeigen und dann auszuführen. Kommentare (mit :: oder REM) werden dadurch natürlich auch nicht mehr ausgegeben, was nur als Nebeneffekt ausfällt.

```
@echo off
:: Verhindert, dass dieser Kommentar angezeigt wird.
```

Ohne @echo off:

```
C:\>::: Dieser Kommentar wird so angezeigt
```

## ! (Ausrufezeichen)

### Syntax:

- Windows XP

```
!VARIABLENAME!
```

### Web-Links:

- ["EnableDelayedExpansion" auf SS64.com \(https://ss64.com/nt/delayedexpansion.html\)](https://ss64.com/nt/delayedexpansion.html)
- [Beispiel zur fehlerhaften Verwendung von "!" inkl. Erklärung \(engl.\): "Batch - Adding Users to Multiple Groups Through For Loop" auf stackoverflow.com \(http://stackoverflow.com/questions/15309422/batch-adding-users-to-multiple-groups-through-for-loop\)](http://stackoverflow.com/questions/15309422/batch-adding-users-to-multiple-groups-through-for-loop)
- [Allgemeine Info zur Allokation von Variablen](#)

## Erklärung:

Zur verzögerten Übersetzung von Variablen. Bewirkt, dass die Variable nicht zur Kompilierzeit sondern erst zur Laufzeit übersetzt wird (setzt die Verwendung von SETLOCAL zur Aktivierung von verzögerter Übersetzung voraus)

## Verwendung:

Beispielsweise zur besser differenzierten Ausführung von For-Schleifen siehe [LINK \(https://ss64.com/nt/delayedexpansion.html\)](https://ss64.com/nt/delayedexpansion.html)

## Beispiele:

Inhalt der Batchdatei

```
Setlocal EnableDelayedExpansion
Set _var=first
Set _var=second& Echo %_var% !_var!
```

Ausgabe (unter XP)

```
first second
```

## : (Doppelpunkt)

---

### Erklärung:

Sprungmarke für ein Unterprogramm bzw. eine Kommentarzeile.

Sprungmarken werden benötigt, wenn mittels der Batchdatei eine Bedingung überprüft und erfüllt bzw nicht erfüllt wird und entsprechend weiter verfahren werden soll.

Mit dem Batchbefehl goto wird die Sprungmarke angesprungen.

### Anmerkung:

Der Doppelpunkt hat auch die Funktion der Manipulation von Variablen, wenn er direkt hinter einer Variablen steht. Siehe Kapitel "Variablen" in dieser Publikation.

### Syntax

:NAMEDERSPRUNGMARKE

Sprungmarken können eine beliebige Länge haben, unter MS-DOS und älteren Windows-Versionen werden allerdings nur die ersten 8 Zeichen beachtet, der Rest wird ignoriert. Kommen in einer Batch also :Sprungmarke1 und :Sprungmarke2 vor, so wird unter Umständen nur die erste beim Aufruf einer der Beiden gefunden. Also besser :ziel1 oder :1st schreiben. Groß- und Kleinschreibung wird nicht unterschieden. Man kann jedoch mithilfe von Anführungszeichen

dafür sorgen, dass der komplette Sprungname beachtet wird (Beispielsweise :`"Sprungname"`). Wenn man den kompletten Code mit Sprungmarken verbindet, kann man zur besseren Unterscheidung verschiedener Sprungmarken eine Leerzeile verwenden.

### Beispiel:

#### Inhalt

```
if exist C:\blabla.txt goto EDITBLA
goto END

:: Kommentarzeile, sofern es erforderlich ist, einen Kommentar zu schreiben
:: Zur Unterscheidung von Sprungmarken verwende ich zwei "::"
:EDITBLA
edit c:\blabla.txt

:END
```

Sofern die Datei `C:\blabla.txt` existiert, wird sie mit `edit` geöffnet, sonst wird das Unterprogramm übersprungen und die Batchdatei bei der Marke `:END` fortgesetzt, also beendet.

Seit die Befehlsweiterungen aktiviert sind, steht in Batchdateien die Sprungmarke `:EOF` zur Verfügung, welche sich unsichtbar am Ende der Batch-Datei befindet.

## :: (doppelter Doppelpunkt)

### Syntax

- Windows 2000, XP

```
::KOMMENTAR
```

### Erklärung:

Der `::` **kann** zur Einleitung eines "Kommentars" verwendet werden, ist aber eigentlich ein Spezialfall der Verwendung vom Doppelpunkt (Label). Eine ausführliche Erklärung zu Kommentaren findet sich beim Befehl REM. Die Verwendung des doppelten Doppelpunkts funktioniert innerhalb von Abfragen und Schleifen nicht - der REM-Befehl hingegen schon. Es ist daher besser, den doppelten Doppelpunkt überhaupt nicht zu verwenden und REM zu bevorzugen.

### Beispiel:

#### Inhalt der Batchdatei

```
::echo Ich bin ein Kommentar
echo Ich bin ein Befehl
```

#### Ausgabe der Batchdatei

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
```

```
C:\>echo Ich bin ein Befehl
Ich bin ein Befehl
```

Die Zeile "Ich bin ein Kommentar" bleibt bei der Ausführung "unberücksichtigt".

### Beispiel, in dem :: nicht funktioniert:

Inhalt der Batchdatei

```
@echo off
if 1==1 (
  :: Kommentar 1
  :: Kommentar 2
  echo OK
)
```

Ausgabe der Batchdatei

```
Das System kann das angegebene Laufwerk nicht finden.
OK
```

## CALL

Mit call kann man eine andere Batch-Datei aufrufen. Sobald diese beendet wurde, wird die ursprüngliche Batchdatei weiter ausgeführt.

Beispiel:

```
REM Call.cmd
@echo off
echo Diese Batchdatei ruft eine andere auf.
call anderedatei.bat
pause
```

```
REM anderedatei.bat
@echo off
echo Die andere Datei
```

Die Ausgabe wenn man die Datei Call.cmd startet:

```
Diese Batchdatei ruft eine andere auf.
Die andere Datei
{Pause}
```

Wenn die Befehlsweiterungen aktiviert sind (Standard ab Windows 2000) kann man auch Sprungmarken aufrufen (und auch Parameter übergeben):

Beispiel:

```
@echo off
REM Diese Batchdatei ruft eine eigene Sprungmarke auf
echo Vor dem Sprung
call :sprungmarke
echo Nach dem Sprung
pause
goto end

:sprungmarke
echo Sprungmarke aufgerufen!
goto :eof
```

```

:: ":EOF" führt nicht zum unsichtbaren Ende der Batch-Datei, wie unten beschrieben,
:: sondern führt die Batch-Datei nach dem Aufruf der Sprungmarke fort

:end
exit

```

## Die Ausgabe:

```

Vor dem Sprung
Sprungmarke aufgerufen!
Nach dem Sprung

```

## Beispiel mit Parameter:

```

@echo off
REM Diese Batchdatei ruft eine eigene Sprungmarke auf
echo Vor dem Sprung
call :sprungmarke meinParameter
echo Nach dem Sprung
pause
goto end

:sprungmarke
echo Sprungmarke aufgerufen und Parameter %1 uebergeben!
goto :eof

:end
exit

```

## Die Ausgabe:

```

Vor dem Sprung
Sprungmarke aufgerufen und Parameter meinParameter uebergeben!
Nach dem Sprung

```

```
goto :eof
```

Dieser Befehl springt automatisch zum Ende der Batchdatei (beendet die aktuelle Prozedur)

## CHCP

Anzeige der aktuellen CodePage oder Setzen einer neuen CodePage (**change codepage**)

```
chcp [nnn]
```

### Beispiele für nnn:

- 437 – Die ursprüngliche Zeichensatztable des IBM-PC
- 720 – Arabisches Alphabet
- 737 – Griechisches Alphabet
- 850 – westeuropäische Sprachen (DOS-Latin-1)
- 857 – Türkisches Alphabet
- 866 – Kyrillisches Alphabet
- 1252 - DOS8 Umlaute werden richtig Codiert

# CLS

---

Mit `cls` (clear screen) wird der Bildschirm gelöscht.

## Syntax

`cls`

Interner Befehl: Ab MS-DOS bis Windows 8

## Beispiel:

### Inhalt

```
@echo off
echo Hier schreibe ich jetzt ganz viel Text.
echo Hier kann ich z.B. hinschreiben, dass ich jemanden mag.
echo.
echo Aber den Text sieht man gleich sowieso nicht mehr ... Hihi!
cls
echo War irgendetwas?
pause>nul
```

### Ausgabe

```
War irgendetwas?
```

# CMD

---

## Syntax:

- Windows 2000, XP, Vista, 7, 8 und 10

```
cmd /a|/u|/q|/d|/e (ON|OFF)|/f (ON|OFF)|/v (ON|OFF)|/c befehl|/s|/k|/y
```

- DOS, Windows 95, 98, ME

entspricht dem Batch-Befehl `COMMAND` (CMD ist unter DOS nicht vorhanden)

## Web-Links:

- [Windows XP Professional Product Documentation \(http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/ntcmds.mspx?mfr=true\)](http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/ntcmds.mspx?mfr=true)

## Erklärung:

Der Befehl `CMD` startet eine neue Instanz des Kommandozeileninterpreters (`cmd.exe`).

## Verwendung:

In Kombination mit dem Befehl `START` öffnet sich die neue Instanz auch in einem neuen Kommandozeilenfenster (ohne `START` öffnet sie sich im gleichen Fenster).

## Beispiele:

### Inhalt der Batchdatei

```
cmd
```

### Ausgabe (unter XP)

```
C:\>cmd
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>
```

### Ausgabe (unter Vista)

```
C:\>cmd
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation. Alle Rechte vorbehalten.

C:\>
```

### Ausgabe (unter Win 7)

```
C:\>cmd
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Alle Rechte vorbehalten.

C:\>
```

### Ausgabe (unter Win 8 Beta)

```
C:\>cmd
Microsoft Windows [Version 6.2.8250]
Copyright (c) 2012 Microsoft Corporation. Alle Rechte vorbehalten.

C:\>
```

### Ausgabe (unter Win 10)

```
C:\>cmd
Microsoft Windows [Version 10.0.10586]
Copyright (c) 2015 Microsoft Corporation. Alle Rechte vorbehalten.

C:\>
```

## COLOR

Mit dem Befehl COLOR kann man die Vorder- und Hintergrundfarbe verändern. Die COLOR Werte bestehen aus zwei HEX-Werten, wobei der erste HEX-Wert für die Hintergrundfarbe und der zweite HEX-Wert für die Vordergrundfarbe steht. Jede Ziffer kann einen der folgenden Werte annehmen:

HEX-Wert	Farbe	HEX-Wert	Farbe
----------	-------	----------	-------

0	Schwarz	8	Dunkelgrau
1	Dunkelblau	9	Blau
2	Dunkelgrün	A	Grün
3	Blaugrün	B	Zyan
4	Dunkelrot	C	Rot
5	Lila	D	Magenta
6	Ocker	E	Gelb
7	Hellgrau	F	Weiß

Der folgende Befehl ergibt z.B. einen blauen Hintergrund mit weißer Schrift. (Ähnlichkeit mit dem Bluescreen in älteren Windows Versionen)

```
COLOR 9F
```

Zum Zurücksetzen der Vorder- und Hintergrundfarbe wird COLOR einfach ohne Argumente aufgerufen.

```
COLOR
```

## COMMAND

---

(Unter Windows 10 nicht mehr anwendbar) **Syntax:**

- DOS, Windows 95, 98, 98 SE, ME

```
command Laufwerk:Pfad Gerät /e /l /u /P /MSG /LOW (/Y (/c|/k) Befehl)
```

### Web-Links:

---

### Erklärung/Verwendung:

Startet einen neuen Kommandointerpreter, dieser kann mit `exit` wieder beendet werden.

### Beispiel:

#### Inhalt

```
command
```

#### Ausgabe (unter Windows 95)

```
C:\WINDOWS>command
```

```
Microsoft(R) Windows 95
(C)Copyright Microsoft Corp 1981-1996.
```

```
C:\WINDOWS>
```

## DATE

---

Gibt das aktuelle Datum aus und ermöglicht dem Benutzer die Änderung des Datums. Wird der Befehl mit dem Parameter /t aufgerufen, so wird nur das aktuelle Datum ausgegeben.

DATE kann auch als Variable benutzt werden, so kann man zum Beispiel mit %date:~6,4% auf das Jahr zugreifen.

Beispiele:

```
z:\>date /T
12.06.2013
```

```
z:\>echo Heute ist der %date%.
Heute ist der 12.06.2013.
```

```
z:\>echo %date:~6,4%
2013
```

Hier werden vom Datum die ersten 6 Zeichen weggelassen und dann vier Stellen angezeigt. Bei time funktioniert das analog.

```
z:\>echo %DATE:~6-%DATE:~3,2-%DATE:~0,2 &REM ISO-Datum für nach Dateiname sortierbarer zeitlicher Reihenfolge
2019-08-22
```

Bei einigen Betriebssystemversionen erfordert das Ändern des Systemdatums administrative Rechte.

XP: Die Ausgabe des Datumsformates ist abhängig von den Einstellungen in den Regions- und Sprachoptionen (Systemsteuerung)

Hinweis: Die Uhrzeit lässt sich mit dem Befehl "TIME" ermitteln.

## ECHO

---

Gibt einen Text aus oder schaltet die Befehlszeilen an/aus. Wenn ein Text ausgegeben wird, können dort auch Variablen angezeigt werden, wie z. B. die Variable %**ver**% (in Windows XP %**os**%).

**Syntax:**

```
echo text |ON|OFF oder alternativ echo. [text]
```

Interner Befehl: Ab MS-DOS bis Windows NT 5.1 (XP)

**Beispiel:**

Inhalt

```
@echo off
echo Die aktuelle Datei heißt %0.
echo Die aktuelle Version Ihrer Befehls-Konsole oder -OS heißt %ver%
```

## Ausgabe

```
Die aktuelle Datei heißt beispiel.bat.
Die aktuelle Version Ihrer Befehls-Konsole oder -OS heißt Windows NT
```

Mit `echo`. können Sie zudem leere Zeilen ausgeben. **Beispiel:**

## Inhalt

```
@echo off
echo Jetzt gibt es 3 Leere Zeilen zu sehen!
echo.
echo.
echo.
echo So! Da waren sie.
```

## Ausgabe

```
Jetzt gibt es 3 Leere Zeilen zu sehen!

So! Da waren sie.
```

## Benutzereingaben mit ECHO

Mit Hilfe des `echo`-Befehls können Sie auch in einem Skript Benutzereingaben simulieren, indem Sie den Pipe-Operator `|` verwenden.

### Beispiel: Uhrzeit anzeigen ohne Nachfrage

Das normale Verhalten des `time`-Befehls ist, die aktuelle Zeit der verwendeten Systemuhr anzuzeigen und in der nächsten Zeile die Eingabe einer neuen Uhrzeit zu erwarten. Drückt man auf Enter, bleibt die Systemzeit unverändert. Will man die Zeit nur anzeigen lassen (z. B. in einer Batchdatei vor und nach einer Befehlsfolge, um zu messen, wie lange der PC dafür braucht), lässt sich die Betätigung der Enter-Taste durch einen entsprechenden `echo`-Befehl ersetzen.

```
echo.|time
```

Über den Pipe-Mechanismus lässt sich darüber hinaus die Zeile „Geben Sie die neue Uhrzeit ein.“ unterdrücken:

```
echo.|time|find /v "neue"
```

Dabei ist die Groß-/Kleinschreibung von "neue" zu beachten oder der Schalter `/I` zu verwenden, denn `find` ist case-sensitiv! Diese Beispiele dienen allerdings nur zur Demonstration, denn die Zeitausgabe wäre auch ohne Pipes (aber erst ab Windows 2000) möglich mit:

```
time /t
```

oder einfach:

```
echo %time%
```

Sehr nützlich ist das echo-Piping auch zur Übergabe von Benutzereingaben, welche von einzelnen Befehlen abgefragt werden.

### Beispiel: Überprüfung einer Festplatte

```
chkdsk c: /f /r
```

kann (da es sich beim Laufwerk C um das Systemlaufwerk handelt) erst nach einem Systemstart ausgeführt werden. Normalerweise müsste der Benutzer deswegen den Systemstart durch Eingabe von "Y" bestätigen. Diese Aktion kann man in einem Batch so abbilden:

```
echo y | chkdsk c: /f /r
```

## ERRORLEVEL

---

### Syntax:

- Windows XP, Windows Vista, Windows 7, Windows 10

```
%errorlevel%
```

### Verwendung:

Zeigt an, ob der letzte Befehl erfolgreich war. Zum Prüfen ob der letzte Befehl ohne Fehler war, kann der folgende Code verwendet werden:

```
IF ERRORLEVEL 1 ...
```

Dann kann im Fehlerfall etwas (...) gemacht werden. Dies funktioniert auch wenn der Errorlevel  $\geq 1$  ist.

### Beispiel

```
del %homepath%\Desktop\Ordner
echo %errorlevel%
```

- del %homepath%\Desktop\Ordner** = Im Homepath (C:\Users\Username) wird im Ordner "Desktop" der Ordner "Ordner" gelöscht.
- echo %Errorlevel%** = Errorlevel wird angezeigt. 1 bedeutet "Befehl fehlgeschlagen", 0 bedeutet "Befehl ausgeführt". 2 bedeutet "Unbekannt".

## ENDLOCAL / (SETLOCAL)

---

### Syntax:

- Windows XP

```
ENDLOCAL
```

**Links:**

- <http://ss64.com/nt/endlocal.html>

**Verwendung:**

Schliesst den Befehl SETLOCAL ab. Siehe [SETLOCAL](#)

**EXIT**

---

Der Befehl `exit` beendet die Abarbeitung der Batchdatei bzw. die Kommandozeilenfenster. Als Parameter kann der an das aufrufende Programm zu übertragende `ReturnCode` angegeben werden.

Beispiel: `exit 255`

**FOR**

---

Ermöglicht die Schleifenbearbeitung.

**Syntax:**

*for Variable in Satz do Befehl [Parameter]*

Interner Befehl: Ab MS-DOS bis Windows NT 6.1 (Windows 7)

**ACHTUNG:**

Die Variable darf nur aus einem Buchstaben bestehen! "%t" ist erlaubt, "%test" nicht! Bei der Verwendung mehrerer Befehle muss zwischen "DO" und der Klammer "(" ein Leerzeichen sein.

Falsch

```
for Variable in Satz do(
```

**RICHTIG**

```
for Variable in Satz do (
  Befehl1
  Befehl2
)
```

**Beispiel:**

Zeigt alle Dateien im Verzeichnis `%temp%` an. Es werden nur Dateien, keine Verzeichnisse angezeigt. Um Verzeichnisse anzuzeigen siehe Liste der FOR-Optionen unten. Der Parameter `/R` bewirkt, dass alle Unterverzeichnisse mit einbezogen werden (Rekursive Schleife).

Inhalt

```
@echo off
for /R %temp% %%f in (*.*) do (
  echo %%f
```

```
)
REM Den Befehl könnte man auch einzeilig schreiben.
pause
```

## Ausgabe

```
(Alle Temp-Dateien)
Bitte beliebige Taste drücken...
```

## Zählschleifen

Mit solchen Schleifen kann man Aktionen eine bestimmte Anzahl oft ausführen. Dazu muss man den Parameter /L angeben.

Syntax: `for /L {Variable} IN (Startzahl, Schrittweite, Endzahl) DO (Aktion)`

```
REM Schreibe Text 5 Mal
for /L %%N IN (1, 1, 5) DO echo Nummer %%N
```

## Ausgabe:

```
C:\>for /L %%N IN (1, 1, 5) DO echo Nummer %%N

C:\>echo Nummer 1
Nummer 1

C:\>echo Nummer 2
Nummer 2

C:\>echo Nummer 3
Nummer 3

C:\>echo Nummer 4
Nummer 4

C:\>echo Nummer 5
Nummer 5
```

## verschachtelte Zählschleife:

```
for /L %%N IN (1, 1, 5) DO
for /L %%N IN (1, 1, %%N)DO echo Nummer %%N
```

## Ausgabe:

```
C:\>for /L %%N IN (1 1 5) DO (for /L %N IN (1 1 %N) DO echo Nummer %N )

C:\>(for /L %%N IN (1 1 1) DO echo Nummer %N )

C:\>echo Nummer 1
Nummer 1

C:\>(for /L %%N IN (1 1 2) DO echo Nummer %N )

C:\>echo Nummer 1
Nummer 1

C:\>echo Nummer 2
Nummer 2

C:\>(for /L %%N IN (1 1 3) DO echo Nummer %N )
```

```

C:\>echo Nummer 1
Nummer 1

C:\>echo Nummer 2
Nummer 2

C:\>echo Nummer 3
Nummer 3

C:\>(for /L %%N IN (1 1 4) DO echo Nummer %N )

C:\>echo Nummer 1
Nummer 1

C:\>echo Nummer 2
Nummer 2

C:\>echo Nummer 3
Nummer 3

C:\>echo Nummer 4
Nummer 4

C:\>(for /L %%N IN (1 1 5) DO echo Nummer %N )

C:\>echo Nummer 1
Nummer 1

C:\>echo Nummer 2
Nummer 2

C:\>echo Nummer 3
Nummer 3

C:\>echo Nummer 4
Nummer 4

C:\>echo Nummer 5
Nummer 5

```

## Weitere Möglichkeiten der FOR-Schleife:

### syntax-FOR-Files

```
FOR %%parameter IN (set) DO command
```

### syntax-FOR-Files-Rooted at Path

```
FOR /R drive:path %%parameter IN (set) DO command
```

### syntax-FOR-Folders

```
FOR /D %%parameter IN (folder_set) DO command
```

### syntax-FOR-List of numbers

```
FOR /L %%parameter IN (start,step,end) DO command
```

### syntax-FOR-File contents

```
FOR /F ["options"] %%parameter IN (filenameset) DO command
```

```
FOR /F ["options"] %%parameter IN ("Text string to process") DO command
```

## syntax-FOR-Command Results

```
FOR /F ["options"] %%parameter IN ('command to process') DO command
```

**Beispiel:** Sucht im Ordner C:\Windows\Temp rekursiv nach Dateien mit dem Namen //temp.dat// und gibt die Liste aus. Die Option "tokens=\*" ist notwendig, damit die Ausgabe zeilenweise gelesen wird und auch Pfade mit enthaltenem Leerzeichen ausgegeben werden können.

```
for /F "tokens=*" %%f in ('dir /S /b C:\Windows\Temp\temp.dat') do (
  echo "%%f".
)
```

### Zählvariablen in Zeichenketten einbetten

Um die Zählvariable %%f (%f auf der Kommandozeile) in einer Zeichenfolge zu verwenden, wird einfach die Variable in dem String eingebettet

```
FOR %%f IN (A B C D E) DO (
  echo mitten%%fdrinnen
)
```

### Listen erzeugen

```
@setlocal ENABLEDELAYEDEXPANSION
@set Dateinamen=
@for %%d in (
  "Langer Name 3"
  "Langer Name 2"
  "Langer Name 3"
) do @set Dateinamen=!Dateinamen! %%d
echo %Dateinamen%
```

Ist rechenaufwendiger, aber weniger fehleranfällig als ein mehrzeiliges SET mit ^-maskierten Zeilenumbrüchen, wo ein ^ am Ende der Liste ein Return mitten im späteren Befehlsaufruf auslösen würde.

## GOTO

Mit dem Batchbefehl goto wird eine Sprungmarke : (s.o.) angesprungen.

### Syntax

```
goto NAMEDERSPRUNGMARKE
```

### Beispiel

Siehe unter : (Doppelpunkt).

## IF

Der IF-Befehl ermöglicht eine einfache Verzweigung und wird oft zusammen mit dem GOTO-Befehl eingesetzt. IF ermöglicht hierbei sowohl die Prüfung auf eine Gleichheit als auch auf das Vorhandensein von Dateien.

### Beispiel 1:

```
@echo off
IF exist c:\temp\my.log echo.>c:\temp\my.log
echo.Log Datei erstellt>>c:\temp\my.log
```

Beispiel 1 prüft, ob eine Logdatei vorhanden ist und erstellt ggf. eine Neue.

### Beispiel 2:

```
@echo off
IF "%COMPUTERNAME%" == "Bastie" GOTO WAHR
REM hier landet man wenn der if-Ausdruck falsch ist
GOTO WEITER
:WAHR
REM hier landet man wenn der if-Ausdruck wahr ist
echo Willkommen Zuhause
REM Jetzt wird der if Zweig verlassen
GOTO WEITER

:WEITER
echo.Have a nice Day!
```

### Beispiel 3:

```
IF "%COMPUTERNAME%" == "Bastie" (
    echo Willkommen zu Hause!
) ELSE (
    echo Du bist auf Computer: %COMPUTERNAME%
)
echo. Schönen Tag noch!
```

Beachten Sie, bei der Prüfung von Umgebungsvariablen niemals

```
IF %Umgebungsvariable% == Prüfwert ...
```

zu schreiben, wenn die Umgebungsvariable nicht gesetzt ist; Sie erhalten sonst einen Syntaxfehler. Der Parameter /i unterbindet eine Differenzierung der Groß-/Kleinbuchstaben.

### ACHTUNG:

Bei der Verwendung mehrerer Befehle muss zwischen *Bedingung* und der Klammer "(" ein Leerzeichen sein.

Falsch

```
IF Bedingung(
```

Richtig

```
IF Bedingung (
    Befehl1
    Befehl2
)
```

## Syntaxvergleiche

IF <NOT> Variable1==Variable2

IF %Variable% EQU %Variable2% (Befehl) An die Stelle von EQU kann jede der Optionen gesetzt werden.

**NOT** Der Befehl wird nur ausgeführt, wenn die Bedingung NICHT Wahr ist. Optional.

**==** ist gleich

**EQU** ist gleich

**NEQ** nicht gleich

**LSS** kleiner als

**LEQ** kleiner als oder gleich

**GTR** größer als

**GEQ** größer als oder gleich

## Hinweis zu UND- bzw. ODER-Verknüpfung

Eine UND- bzw. ODER-Verknüpfung von zwei Bedingungen scheint nicht direkt möglich zu sein. Beim Vergleichen von Strings hilft es aber eventuell, wenn man die beiden Strings miteinander verkettet.

### Beispiel

```
set A=true
set B=false
if "%A;%B%"=="true;true" (
    echo A und B sind beide TRUE
) else (
    echo mindestens eines von beiden - A oder B - ist ungleich TRUE
)
```

Als Workaround können mehrere aufeinanderfolgende IFs zu einer UND- bzw. ODER-Verknüpfung kombiniert werden. Bei einer ODER-Verknüpfung wird der Code ausgeführt, sobald eine der Bedingungen wahr ist. Wenn alle Bedingungen geprüft wurden und keine erfolgreich war, werden die Befehle im ELSE-Zweig ausgeführt.

```
set A=true
set B=false
if "%A%"=="true" goto :WAHR // Diese Zeile ist doch erfüllt, also sollte der in :WAHR springen
if "%B%"=="true" goto :WAHR
REM keine der Bedingungen ist zu :WAHR gesprungen, wir sind also im ELSE-Zweig
REM hier waere :FALSCH

echo Weder A noch B ist TRUE
goto :eof

:WAHR
echo A oder B ist TRUE
```

Für ein UND wird in den ELSE-Zweig gesprungen ( :FALSCH ) sobald eine der Bedingungen **nicht** zutrifft. Nur wenn alle Bedingungen zutreffen wird der Code ausgeführt.

```
set A=true
set B=false
if "%A%"NEQ"true" goto :FALSCH
if "%B%"NEQ"true" goto :FALSCH
```

```

REM wird sind durch die IFs gekommen, also hat keine der Bedingungen angeschlagen.
REM hier waere die :WAHR-Sprungmarke

echo A und B sind beide TRUE
goto :eof

:FALSCH
echo A oder B (oder beide) sind FALSE

```

Die am weitesten verbreitete, kurze und gut lesbare UND-Variante ist, zwei oder mehrere IF-Bedingungen nacheinander zu schreiben.

```

set P1=Frankreich
set P2=Spanien

if "%P1%"=="Frankreich" if "%P2%"=="Spanien" echo Die Nachbarländer von Andorra sind %P1% UND %P2%

```

Diese beiden Beispiele lassen sich einfach durch Kopieren der "IF..."-Zeile um beliebig viele Bedingungen erweitern. Ein Mischen von UND- und ODER-Verknüpfungen ist leider nicht ohne weiteres möglich. Außerdem dürfen Parameter niemals leer sein - deshalb sollten Variablen z. B. in Anführungszeichen eingeschlossen werden.

## KEYB

---

Lädt Tastatortreiber. `keyb gr,,c:\dos\keyboard.sys` lädt den deutschen Tastatortreiber (keyboard.sys muss sich im Verzeichnis `c:\dos\` befinden)

## PAUSE

---

Unterbricht die Abarbeitung der Batchdatei und wartet auf einen Tastendruck.

### Syntax

`pause`

Interner Befehl: Ab MS-DOS bis Windows NT 5.1 (XP)

### Beispiel 1:

#### Inhalt

```

@echo off
echo Willkommen zur Batchdatei %0 !!!
echo.
echo Die Batchdatei wird auf einer neuen Seite fortgeführt.
pause
cls
echo Hier fängt meine Batchdatei an...
pause

```

#### Ausgabe

```

Willkommen zur Batchdatei beispiel.bat !!!

```

```
Die Batchdatei wird auf einer neuen Seite fortgeführt.
Bitte beliebige Taste drücken...
```

*(Neue Seite)*

```
Hier fängt meine Batchdatei an...
Bitte beliebige Taste drücken...
```

## Beispiel 2:

### Inhalt

```
@echo off
echo Diese Zeile wird angezeigt anstelle von "Bitte beliebige Taste drücken..."
pause > NUL
```

### Ausgabe

```
Diese Zeile wird angezeigt anstelle von "Bitte beliebige Taste drücken..."
```

Um eine Pause für eine bestimmte Zeitdauer vorzugeben, kann der Befehl `sleep` verwendet werden.

## PROMPT

---

- **Syntax:**

```
prompt [text]
```

- **Erklärung:**

Legt das Aussehen der Eingabezeile mit Hilfe unterschiedlicher Parameter [1] (<http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/ntcmds.mspx?mfr=true>) fest.

- **Beispiele**

```
prompt Guten Tag! Ausgabe: Guten Tag!(Hier kommt die Eingabe des Benutzers)
```

- **Hinweis:**

Der Prompt kann auch permanent über eine Umgebungsvariable geändert werden:

```
set PROMPT=$p$g
```

Sollte der Befehl alleine und ohne Funktion eingegeben werden, so wird die Wirkung aufgehoben.

## PUSHD / POPD

---

`pushd` wechselt zum angegebenen Pfad und speichert den aktuellen Pfad bis zum Aufruf von `popd`.

`popd` wechselt zum gespeicherten Pfad.

Die Befehle können geschachtelt werden.

## Syntax

```
pushd pfad
```

```
popd
```

## Beispiel:

```
C:\WINDOWS>pushd c:\temp
C:\temp>pushd c:\
C:\>popd
C:\temp>popd
C:\WINDOWS>
```

## REM

---

### Syntax:

```
rem [comment]
```

### Erklärung:

REM leitet einen Kommentar ein. Die Zeile wird ignoriert, beachten Sie jedoch, dass REM von einem Leerzeichen / Tabulator gefolgt werden muss, sonst wird die Ausführung der Batchdatei unmittelbar beendet. Alternativ können zwei Doppelpunkte hintereinandergeschrieben werden :: um einen Kommentar anzuführen (hier ist ein Leerzeichen nicht notwendig). Alternativ können auch Sprungmarken : oder der Echobefehl echo verwendet werden.

### Beispiele:

```
REM kill iexplore.exe
::kill iexplore.exe
:kill iexplore.exe
echo kill iexplore.exe
```

### Web-Links:

- [Windows XP Professional Product Documentation \(Syntax-Link\) \(http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/rem.mspx?mfr=true\)](http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/rem.mspx?mfr=true)
- [Unterschiede alternativer Verwendungen \(http://superuser.com/questions/82231/how-do-i-do-comments-at-a-windows-command-prompt\)](http://superuser.com/questions/82231/how-do-i-do-comments-at-a-windows-command-prompt)
- [Unterschiede alternativer Verwendungen \(http://www.obrienpc.net/dos/y-rem.html\)](http://www.obrienpc.net/dos/y-rem.html)
- [Zu den Unterschieden von REM oder ":" \(http://www.computerhope.com/forum/index.php?topic=107102.0\)](http://www.computerhope.com/forum/index.php?topic=107102.0)

## RENAME

---

Mit dem rename-Befehl kann man Dateien umbenennen.

**Syntax:**

- MS-DOS und Windows: REN
- seit Windows 98 auch RENAME

```

RENAME [Laufwerk:][Pfad]Dateiname1 Dateiname2
REN [Laufwerk:][Pfad]Dateiname1 Dateiname2

```

**SET**

---

**Syntax:**

```
set [[/a [expression]] [/p [variable=]] string]
```

**Verwendung:**

Wird hauptsächlich verwendet, um einer Variablen einen Wert zuzuweisen. Siehe [Batchbefehle \(Variablen und Set-Befehl\)](#)

Ohne Parameter gibt set eine Liste aller Umgebungsvariablen aus.

Der Parameter /p kann dazu verwendet werden der Variable eine Benutzereingabe zuzuweisen.

Der Parameter /a kann dazu verwendet werden, um mit den Variablen Rechenoperation durchzuführen.

**Beispiel zur Wertezuweisung:**

Inhalt Batchdatei:

```

set VARIABLENNAME=test
echo %VARIABLENNAME%
set VARIABLENNAME=test2
echo %VARIABLENNAME%

```

Ausgabe (gekürzt):

```

c:\>set VARIABLENNAME=test
c:\>echo %VARIABLENNAME%
test

c:\>set VARIABLENNAME=test2
c:\>echo %VARIABLENNAME%
test2

```

```
C:\>set /p test=Dies ist ein Test:
```

**SETLOCAL / (ENDLOCAL)**

---

**Syntax:**

```

SETLOCAL
SETLOCAL EnableDelayedExpansion | DisableDelayedExpansion
SETLOCAL EnableExtensions | DisableExtensions

```

**Links:**

- <http://ss64.com/nt/setlocal.html>

Siehe auch [Batchbefehle \(Variablen und Set-Befehl\)](#)

## TIME

---

Gibt die aktuelle Zeit aus und ermöglicht dem Benutzer die Änderung der Uhrzeit. Wird der Befehl mit dem Parameter /t aufgerufen, so wird nur die aktuelle Zeit ausgegeben. `time` kann auch als Variable benutzt werden, so kann man zum Beispiel mit `%time:~0,5%` die ersten 5 Zeichen übernehmen.

Beispiele:

```
C:\>time
Aktuelle Zeit: 10:03:04,63
Geben Sie die neue Zeit ein:
c:\>echo %time%
10:03:04,63
c:\>echo %time:~0,5%
10:03
```

Möchte man die aktuelle Zeit für einen Dateinamen verwenden, stört der Doppelpunkt, der in Dateinamen nicht erlaubt ist. Ich kenne zwei Lösungen:

```
c:\>echo %TIME:=% &REM
173117,33
```

```
C:\>echo %time:~0,2%.%time:~3,2%
18.06
```

Bei einigen Betriebssystemversionen erfordert das Ändern der Systemzeit administrative Rechte.

## TITLE

---

Dieser Befehl ändert die Fensterüberschrift der Eingabeaufforderung. (Falls der Befehlsinterpreter mit Administrator-Rechten gestartet wurde, also "Administrator:" im Titel steht, kann nur der Text nach dem `:` geändert werden.)

```
title Beispiel
```

>, >>

---

### Erklärung

Dieser Befehl überschreibt, beziehungsweise hängt etwas an eine vorhandene Datei an.

**Beispiel >:**

Inhalt:

```
@echo off
echo Die Datei wird nun überschrieben.
echo Die Datei wurde überschrieben! > %homepath%\Desktop\Beispiel.bat
```

### Ausgabe:

```
Die Datei wird nun überschrieben.
```

### In Beispiel.bat:

```
Die Datei wurde überschrieben!
```

### Beispiel >>:

#### Inhalt:

```
@echo off
echo Nun wird etwas an die Datei angehängt.
echo Dies wurde an die Datei angehängt! >> %homepath%\Desktop\Beispiel.bat
```

### Ausgabe:

```
Nun wird etwas an die Datei angehängt.
```

### In Beispiel.bat:

```
Die Datei wurde überschrieben!
Dies wurde an die Datei angehängt!
```

## TIMEOUT

---

**Erklärung** TIMEOUT hält eine Stapeldatei für eine anzugebende Zahl von Sekunden an, gibt aber dem Anwender die Möglichkeit, die Wartezeit mit einem Tastendruck vorzeitig zu beenden. Der Befehl ist seit Windows 7 verfügbar.

### Beispiel

#### Inhalt:

```
timeout /t <TimeoutInSekunden> [/nobreak]

Parameter /t -1 für unendlich, bis zu 99999
Parameter /nobreak ignoriert Eingaben des Benutzers, abrechnen dann nur über [strg]+[c] möglich (kann, wenn benötigt
```

### Ausgabe:

```
C:\>timeout 5 /nobreak
Gewartet wird 5 Sekunden. Drücken Sie STRG+C, um den Vorgang zu beenden...

C:\>timeout 5
Gewartet wird 5 Sekunden. Weiter mit beliebiger Taste...
```

---

Um den Kommentar des Timeouts auszublenden, muss " > nul " angehängt werden.

### Beispiel

```
C:\>timeout 5
```

Ausgabe: Es wird noch 5 Sekunden gewartet.

```
C:\timeout 5 > nul
```

Ausgabe: (keine Ausgabe)

## Batch Operatoren

### &

---

Befehlsverkettung: mehrere Batch-Befehle in einer Zeile können hintereinander ausgeführt werden

#### Syntax

```
befehl1 & befehl2
```

Stapelanweisung: Ab ? bis Windows 7 (geht auch unter Windows 10)

#### Beispiel:

Inhalt

```
echo Hallo! & echo und Tschüss!
```

Ausgabe

```
Hallo!  
und Tschüss!
```

---

Befehlsverkettung mittels "pipe": der zweite Befehl bekommt die Ausgabe des ersten Befehls als Eingabe

#### Syntax

```
befehl1 | befehl2
```

Stapelanweisung: Ab DOS (nur für more als zweiter Befehl); allgemein verwendbar ab WindowsNT bis Windows 7

#### Beispiel:

Inhalt

```
type test.txt | more
```

## Ausgabe

```
Zeile 1
Zeile 2
...
Zeile 24
-- More --
```

(nach Drücken einer beliebigen Taste werden die nächsten Zeilen angezeigt)

## &&

---

bedingte Befehlsverkettung: der zweite Befehl wird nur ausgeführt, wenn der erste Befehl erfolgreich war

### Syntax

```
befehl1 && befehl2
```

Stapelanweisung: Ab ? bis Windows 7

### Beispiel:

#### Inhalt

```
copy test.txt test2.txt && echo Kopieren erfolgreich!
```

## Ausgabe

```
Kopieren erfolgreich!
```

(falls test.txt existiert und nach test2.txt kopiert werden konnte)

## ||

---

bedingte Befehlsverkettung: der zweite Befehl wird nur ausgeführt, wenn der erste Befehl fehlschlug

### Syntax

```
befehl1 || befehl2
```

Stapelanweisung: Ab ? bis Windows 7 (geht auch unter Windows 10)

### Beispiel:

#### Inhalt

```
copy test.txt test2.txt || echo Fehler beim Kopieren!
```

## Ausgabe

```
Fehler beim Kopieren!
```

(falls test.txt nicht existiert oder test2.txt nicht geschrieben bzw. überschrieben werden konnte)

als Erweiterung:

```
copy test.txt test2.txt && echo Kopieren erfolgreich || echo Fehler beim Kopieren
```

## Ausgabe

```
falls Kopieren erfolgreich war:      Kopieren erfolgreich
falls Kopieren nicht erfolgreich war: Fehler beim Kopieren
```

---

<

## Umleitung der Eingabe

### Syntax

befehl < file

### Beispiel:

### Inhalt

```
echo Hallo! > tmp.txt
set /P v= < tmp.txt
echo %%
```

## Ausgabe

```
Hallo!
```

(set /P v= würde eine Zeile vom Bediener erwarten; durch die Umleitung wird diese Zeile von der Datei tmp.txt gelesen, die zuvor mittels Umleitung der Ausgabe (siehe unten) befüllt wurde.)

---

>

Umleitung der (Standard-)Ausgabe zu einem anderen Ziel. Wenn das Ziel eine Datei ist, wird diese neu angelegt (falls die Datei schon existiert, wird sie zuvor gelöscht) Es kann aber auch nach NUL(Die Ausgabe verschwindet) oder CON(Bildschirm) umgeleitet werden

### Syntax

befehl > file

oder auch mit vorangestellter Umleitung (praktisch wenn bei der Ausgabe einzelne " vorkommen)

> file befehl

**Beispiel:**

## Inhalt

```
echo Hallo! > tmp.txt
type tmp.txt
```

## Ausgabe

```
Hallo!
```

Neben der Standard-Ausgabe (**stdout**) gibt es noch eine Fehlerausgabe (**stderr**). Normalerweise landen die Standard-Ausgabe und die Fehler-Ausgabe am Bildschirm, wodurch der Unterschied nicht auffällt. Wenn die Standard-Ausgabe auf eine Datei umgelenkt wird, landen Fehlerausgaben immer noch auf dem Bildschirm, was oft erwünscht ist. Falls nicht, kann auch die Fehler-Ausgabe in eine Datei umgeleitet werden, und zwar mit `2>` .

**Beispiel:**

```
dir >stdout.txt 2>stderr.txt
```

Schreibt den Output des `dir` Befehles in die Datei *stdout.txt*, da kein Fehler aufgetreten ist.

```
dir DieseDateiExistiertNicht.txt >stdout.txt 2>stderr.txt
```

Schreibt die Fehlermeldung des `dir` Befehles in die Datei *stderr.txt*, da die Datei *DieseDateiExistiertNicht.txt* nicht existiert und somit ein Fehler auftritt.

Wenn die Fehlerausgabe in die gleiche Datei wie die Standardausgabe umgeleitet werden soll, hängt man noch `2>&1` an.

**Beispiel:**

```
type EineDatei.txt > UmgeleiteteAusgabe.txt 2>&1
```

Der Inhalt der Datei *EineDatei.txt* wird in die Datei *UmgeleiteteAusgabe.txt* kopiert (wenn *EineDatei.txt* existiert).

```
type EineDateiDieNichtExistiert.txt > UmgeleiteteAusgabe.txt 2>&1
```

Wenn die Datei *EineDateiDieNichtExistiert.txt* nicht existiert, dann wird die Fehlermeldung (**stderr**) nicht im Dos-Fenster ausgegeben, sondern wird auch in die Datei *UmgeleiteteAusgabe.txt* geschrieben.

>>

---

Umleitung der (Standard-)Ausgabe mit Anhängen des Textes (falls die Datei schon existiert; sonst wird die Datei wie bei `>` (siehe oben) angelegt)

## Syntax

befehl >> file

Stapelanweisung: Ab ? bis Windows 7

## Beispiel:

### Inhalt

```
echo Hallo! > tmp.txt
echo Haalooo!! >> tmp.txt
type tmp.txt
```

### Ausgabe

```
Hallo!
Haalooo!!
```

Hier gilt analoges für die Fehler-Ausgabe (siehe Operator ' > ' oben).

# Erweiterungen unter Windows NT

## Variablen Kurzübersicht

---

Diese Seite (in diesem Abschnitt eingefügt und überarbeitet) enthielt ursprünglich Änderungen des Kommandozeileninterpreters von DOS auf Windows NT, und beschrieb mehr oder weniger die Erweiterungen für Standard-Befehle.

## Scriptvariablen und Systemvariablen

Es gibt zweierlei Arten von Variablen: Scriptvariablen und Systemvariablen. Die Scriptvariablen werden innerhalb von Schleifen eingesetzt, beginnen mit % und einem frei wählbaren Zeichen, oder einer Zeichenfolge. Systemvariablen müssen von % begonnen und von % abgeschlossen werden. Die Variable %0 hat stets den Namen der aktuellen Datei. Die Variable %ver% hat immer den Namen der aktuellen Betriebssystemversion. Unter Windows XP muss %os% statt %ver% verwendet werden.

Gibt man unter Windows Vista und 7 den Befehl echo %os% ein, so gibt der Computer die Version des Systems aus, auf dem das System basiert. Unter Windows Vista und Windows 7 gibt es einen Befehl namens ver, mit dem man die Version direkt angezeigt bekommt. Es gibt zwar keine direkte Variable bei Vista und Windows 7, die das Betriebssystem definiert, jedoch kann man das mit folgendem Trick umgehen:

```
@echo off           ' Autobefehlsanzeige ausgeschaltet
ver > TMP.dat       ' speichert die Ausgabe des Befehls in der Datei TMP.dat
Set /p ver= < TMP.dat ' definiert die Variable des Betriebssystems. (NUR TEMPORÄR !)
echo %ver%         ' gibt nun die definierte Variable aus.
pause>nul          ' pause
```

## Benutzereingaben in Variablen speichern

Um eine Benutzereingabe in eine Variable speichern zu können, wird die Option /p benötigt.

Beispiel:

```
set /p EINGABE=Variable Eingeben:  
echo %EINGABE%
```

## Zeichenketten in Variablen manipulieren (Stringoperationen)

Nützliche Stringoperation für (String)-Variablen:

### Teilstring

#### Syntax

```
var:~n,m
```

Ergibt den Teilstring von var, beginnend mit dem n-ten Zeichen (von links) und einer Länge von m Zeichen. Gezählt wird ab 0, d.h. das erste Zeichen hat die Position 0 und nicht 1. Werden negative Werte verwendet, so wird vom Ende des Strings (von rechts nach links) gezählt.

#### Beispiel:

Inhalt:

```
set str=myfile.bat  
set name=%str:~0,6%  
echo %name%
```

In diesem Beispiel wird der Teilstring von str vom ersten Zeichen an mit einer Länge von 6 Zeichen ausgegeben.

Ausgabe:

```
myfile
```

### Stringsstitution

#### Syntax

```
var:str1=str2
```

Mithilfe der Syntax var:str1=str2 kann die Zeichenkette str1 des Inhalts der Variablen var durch str2 ersetzt werden.

#### Beispiel:

Inhalt:

```
set str="mycommand /p /m file"
echo %str%
set str=%str:/p /m=/t%
echo %str%
```

In diesem Beispiel werden die Optionen /p /m des Befehls mycommand durch die Option /t ersetzt.

Ausgabe:

```
"mycommand /p /m file"
"mycommand /t file"
```

## Mit Variablen rechnen (Arithmetische Operationen)

Rechnungen mit Variablen sind erst mit der Option /a möglich.

Beispiel:

```
set WERT1=2
set WERT2=21
set /a WERT1 = %WERT1*%WERT2%
echo %WERT1%
```

## Variablenerstellung, -zuweisung und -abruf

---

### Wertezuweisung

Mit dem Befehl "set" kann einer Variablen ein Wert zugewiesen werden. z.B. "set meinName=Peter"

### Werteabruf

Um den Wert einer Variable abzurufen oder auszugeben, wird der Variablenname von Prozentzeichen umgeben. Der Befehl "echo %meinName%" würde die Ausgabe "Peter" erzeugen.

### Links mit umfassenden Informationen zur Variablenverwendung und zum Set-Befehl

- <http://commandwindows.com/variables.htm>

## Gültigkeitsbereich von Variablen

---

### Allgemeine Info zum Gültigkeitsbereich von Variablen

Unter DOS und Windows 9x/ME speichern alle Anwendungen und Skripte ihre Umgebungsvariablen in der gleichen Umgebung. Unter Windows NT-basierten Betriebssystemen ist dies etwas anders:

Zunächst gibt es einen systemweiten Bereich. Dieser enthält Variablen, die für alle Benutzer und alle Anwendungen zugänglich sind. Zusätzlich hat jeder Benutzer seinen eigenen Variablenraum. Jede Anwendung, die nun vom System gestartet wird, erhält einen eigenen Variablenraum, der als Kopie aus dem der aufrufenden Anwendung erzeugt wird. Ändert das Programm etwas (Wertänderung, neue Variable), wirkt sich dies nicht auf parallel gestartete Anwendungen oder das System aus. Lediglich Prozesse, die von der Anwendung gestartet werden, erhalten eine Kopie des Anwendungsvariablenraumes. Dadurch ist es nicht möglich, dass Anwendungen Umgebungsvariablen setzen, die dann anschließend in einer Batchdatei verwertbar sind.

Beispiel:

```
@echo off
set a=2
echo %a%
command /c: set a=1
echo %a%
```

Unter Dos würde die Ausgabe lauten:

```
2
1
```

Unter Windows NT/2000/XP:

```
2
2
```

Das heißt nicht, dass unter Windows NT und Nachfolgern der Befehl `set a=1` keinen Effekt hätte, sondern dieser wird in einer geschützten Umgebung ausgeführt, die dann außerhalb dieser Umgebung keinen Effekt mehr hat. Für die aufgerufene Variante von `command` ist `a` nach dem `set`-Befehl sehr wohl gleich 1.

## Gültigkeitsbereich setzen (setlocal)

Mit dem Befehl `setlocal` lässt sich der Gültigkeitsbereich von Variablen kontrollieren. Für weitere Informationen helfen folgende Links:

- <http://ss64.com/nt/setlocal.html>
- <http://technet.microsoft.com/en-us/library/bb491001.aspx>

Noch eine Änderung Variable betreffend. ([https://de.wikibooks.org/wiki/Batch-Programmierung:\\_Programmierungshilfen#Probleme\\_mit\\_Variablen](https://de.wikibooks.org/wiki/Batch-Programmierung:_Programmierungshilfen#Probleme_mit_Variablen))

## Umgebungsvariablen (Windows Environment Variables)

Umgebungsvariablen werden verwendet, um Variablen anwendungsübergreifend (und skriptübergreifend) zu speichern.

- [Umgebungsvariablenreferenz auf ss64.com \(https://ss64.com/nt/syntax-variables.html\)](https://ss64.com/nt/syntax-variables.html)

- [Weitergehende allgemeine Informationen zu Umgebungsvariablen auf de.wikipedia.org \(http://de.wikipedia.org/wiki/Umgebungsvariable\)](http://de.wikipedia.org/wiki/Umgebungsvariable)

## Weitere "Variablenarten"

Parameter lassen sich bedingt mit Variablen vergleichen und werden zur Werteübergabe beim Aufrufen von Batchdateien verwendet. Auch bei For-Schleifen wird eine Art Parametervariable verwendet.

### Übergabeparameter

#### Syntax:

- Windows XP

```
batchdateiname.cmd PARAMETER1 PARAMETER2 PARAMETER3 ...
```

#### Web-Links:

- [Windows XP Professional Product Documentation \(http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/percent.mspx?mfr=true\)](http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/percent.mspx?mfr=true)
- <http://www.administrator.de/forum/test-cmd-parameter-76928.html>
- <http://ss64.com/nt/syntax-args.html>

#### Erklärung/Verwendung/Beispiele:

Mit Hilfe von Parametern beim Aufrufen einer Batchdatei (z.B. "c:\>batchskript.cmd parameter1 parameter2") Zeichenketten als eine Art Variablen übergeben werden, die dann in der Batchdatei weiterverwendet werden können.

Unter Windows NT lassen sich die Parameter im Gegensatz zu DOS folgendermaßen erweitern (für 1 setze man den jeweiligen Parameter ein):

```

%~1  Anführungszeichen (") werden entfernt
%~f1 vollständige Pfadbezeichnung
%~d1 Laufwerksbuchstabe
%~p1 Pfad (ohne Laufwerksbuchstabe)
%~n1 Dateiname
%~x1 Dateinamenserweiterung
%~s1 Pfad nur mit kurzen Verzeichnis/Dateinamen (8.3-Konvention (https://de.wikipedia.org/wiki/8.3))
%~a1 Dateiattribute
%~t1 Datums- und Uhrzeitangaben der Datei (Geändert am)
%~z1 Größe der Datei

```

Die Angaben können auch kombiniert werden, z.B. %~dpo ist Laufwerksbuchstaben + Pfad der aktuellen Batchdatei.

### Parametervariablen in For-Schleifen

Für For-Schleifen werden auch eine Art von Parameter/Variable verwendet um die Anzahl der Schleifendurchläufe zu kontrollieren.

## Links:

- <http://ss64.com/nt/for.html> For-Schleife auf ss64.com
- <http://ss64.com/nt/syntax-args.html> Parameter auf ss64.com (siehe Abschnitt "FOR parameters")

## Stringoperationen (Manipulation von Strings in Variablen)

---

Mit folgender Syntax lassen sich Teile aus einer Zeichenkette extrahieren:

```
set str=Hallo
set str=%str:~1%
echo.%str%
```

Die angegebene positive Zahl gibt an, wie viele Zeichen links übergangen werden sollen. Dieser Ausdruck liefert daher den Teilstring "allo" von "Hallo".

```
set str=Hallo
set str=%str:~-4%
echo.%str%
```

Wird eine negative Zahl eingesetzt, so gibt ihr Betrag an, wie viele Zeichen von rechts genommen werden sollen. Dieser Ausdruck liefert daher ebenfalls "allo".

Über die Angabe einer zweiten Zahl kann definiert werden, was mit dem Rest nach Auswertung der ersten Zahl entsprechend obiger Regeln geschehen soll. Während die erste Zahl die Zeichenkette links beschneidet, beschneidet die zweite Zahl sie rechts und zwar nach folgenden Regeln:

```
set str=Hallo
set str=%str:~1,2%
echo.%str%
```

Eine positive zweite Zahl gibt an, wie viele Zeichen ab links vom Rest genommen werden sollen. Dieser Ausdruck liefert daher "al".

```
set str=Hallo
set str=%str:~1,-2%
echo.%str%
```

Wird eine negative zweite Zahl eingesetzt, so gibt ihr Betrag an, wie viele Zeichen vom Rest ab rechts übergangen werden sollen. Dieser Ausdruck liefert daher ebenfalls "al".

Möchte man beispielsweise nur einen Teil der Zeichenkette ab links, so ist nach diesen Regeln als erste Zahl eine Null anzugeben:

```
set str=Hallo
set str=%str:~0,2%
echo.%str%
```

Dieser Ausdruck liefert daher "Ha".

Mehr zu Stringmanipulation ([http://www.dostips.com/DtTipsStringManipulation.php#\\_Toc135152735](http://www.dostips.com/DtTipsStringManipulation.php#_Toc135152735))

## Arithmetische Operationen (set /a)

Seit Windows NT 4 kann man mit dem set-Befehl auch rechnen.

Ein Beispiel:

```
set /a 1+2
3
```

Man kann das Ergebnis auch in Variablen schreiben:

```
set /a n=5+5
10
echo %n%
10
```

- CMD nimmt Folgendes entgegen (Auszug aus der Hilfe):

```
( )           - Gruppierung
! ~ -        - unäre Operatoren
* / %        - arithmetische Operatoren (% bedeutet Modulo)
+ -          - arithmetische Operatoren
<< >>        - logische Verschiebung
&            - bitweise UND
^            - bitweise exklusives ODER
|            - bitweise ODER
= *= /= %= += -= - Zuordnung
&= ^= |= <<= >>=
,            - Trennzeichen für Ausdrücke
```

- Mit Klammern:

```
set /a (1+1)*(4-3)
2
```

Modulo-Zeichen muss zweimal hintereinander geschrieben werden.

```
set /a 11 %% 5
1
```

## Ein- und Ausgaben in Variablen speichern ("set /p")

### Benutzereingaben

Seit Windows 2000 kann man mit dem set-Befehl auch Eingaben abfragen und einer Variable zuweisen:

Syntax:

```
set /p {variablenname}={Angezeigter Text}
```

## Beispiel

```
@echo off
set var=
set /p var=Eingabe:
echo Sie haben %var% eingegeben.
pause
```

## Ausgabe:

```
Eingabe: Test
Sie haben Test eingegeben.
{Pause}
```

Wichtig: Gibt der Benutzer nichts ein, so bleibt die Umgebungsvariable unverändert! Also immer initialisieren!

## Prüfung, ob der Benutzer Text eingegeben hat:

```
@echo off
set var=
set /p var=Eingabe:
if not defined var (
    echo Bitte geben Sie etwas ein!
    set var=nichts
    REM Könnte auch was ganz anderes sein
)
echo Sie haben "%var%" eingegeben.
pause
```

## Ausgabe:

```
Eingabe: {leer}
Bitte geben Sie etwas ein!
Sie haben "nichts" eingegeben.
{pause}
```

## Umgeleitete Ausgaben

Eine Möglichkeit der Umleitung einer Ausgabe besteht über den set /p Befehl:

## Syntax:

```
ECHO {Befehl}> {Dateiname}
SET /P {Variablenname}=<{Dateiname}
```

## Beispiel

```
@ECHO OFF
ECHO %date% > _.txt
SET /P tmp=<_.txt
ECHO Das Datum heute ist: %tmp%
```

## Sonderzeichen verwenden

Manchmal will man Zeichenketten mit Sonderzeichen verwenden, die als Teil eines Befehls verstanden werden – beispielsweise % < | >. In diesem Fall muss man das betreffende Zeichen „maskieren“, sodass es nicht als Teil des Befehls, sondern als „echtes“ Zeichen interpretiert wird. Dazu werden Escape-Zeichen verwendet, beispielsweise:

```
%      %%
<      ^<
|      ^|
>      ^>
```

Eine gute Zusammenstellung, wie man mit diesen Escape-Charakteren umgeht, findet sich unter [Escape Characters \(http://www.robvanderwoude.com/escapechars.php\)](http://www.robvanderwoude.com/escapechars.php).

# Programmierungshilfen

## Ändern des Editors zum Bearbeiten von Batchdateien

Wenn man im Windows Explorer mit der Rechten-Maus-Taste (RMT) auf eine \*.bat klickt, so werden einem die Befehle *Öffnen* und *Bearbeiten* angeboten.

**Öffnen:** führt die Batchdatei aus. Mit dem Befehl

**Bearbeiten:** wird die Batchdatei in den Texteditor *notepad.exe* zum Bearbeiten geöffnet.

Auch wenn Notepad zum Bearbeiten von Batchdateien ausreicht, so möchte man häufig doch die Batchdateien mit einem anderen, komfortableren Editor bearbeiten, der z. B. Syntaxhervorhebung (Syntaxhighlighting) beherrscht.

Um einen anderen Editor (z. B. Syn (<http://syn.sourceforge.net/>)) zu verwenden, muss man in der Registry an der Stelle:

```
HKEY_CLASSES_ROOT\batfile\shell\edit\command
```

den Standard Wert

```
(Standard) = %SystemRoot%\System32\NOTEPAD.EXE %1
```

auf den Startbefehl des entsprechenden Editors ändern. Z. B.

```
HKEY_CLASSES_ROOT\batfile\shell\edit\command | (Standard) = c:\Programme\Editor\syn\syn.exe %1
```

## Datum und Uhrzeit anzeigen

( *der Inhalt des Abschnittes ist stark von der Windows-Lokalisierungseinstellungen abhängig und gilt für die Einstellung "Deutschland"* )

Für Log-Dateien ist es wichtig, dass man die Logeinträge mit Datum und Uhrzeit versehen kann:

```
echo %date:~0% - %time:~0,8% Uhr
```

Ergebnis: **11.09.2010 - 15:59:53 Uhr**

Hierbei steht ":~0,8" für die Angabe der Stellen. Die Syntax ist:<sup>[1]</sup>

```
%variable:~skip[,keep]%
```

Mit "0,8" wird angegeben, dass die Ausgabe der Zeit bei Position 0 beginnen soll und insgesamt 8 Stellen beinhalten soll, denn der erste Wert (0) gibt an, wieviel übersprungen werden soll (skip), der zweite Wert (8), wieviele Stellen behalten werden sollen (keep). Es sollen demnach 0 Stellen übersprungen und 8 Stellen behalten werden und so ist das Ergebnis: 15:59:53 Die maximale Stellenanzahl ist hier 11 (0,11), da die %time% Variable am Ende noch ein Komma und Zehntel- und Hundertstelsekunde beinhaltet.

Eine weitere Anwendung findet so ein Zeitstempel bei der Erstellung der eindeutigen Datei-/Ordernamen:

```
set d=%date%
mkdir %USERNAME%-%d:~0,2%%d:~3,2%%d:~6,2%
```

Wobei man beachten muss, dass

1. die Pseudo-Umgebungsvariablen %date% und %time% von Windows jeweils beim Aufführen eines Codeabschnitts (einer Zeile oder einer for-Schleife) genau einmal ausgewertet und ersetzt werden. Deswegen muss man sie ggf. einmal abfragen, in der Zwischenvariablen speichern und danach nur die Zwischenvariablen verwenden ( die Konstrukte wie %time:~0,2%%time:~3,2%%time:~6,2% vermeiden - zumindest, wenn sie sich auf mehrere Zeilen verteilen), weil die mehrmalige Benutzung von der %date% und %time% in seltenen unglücklichen Fällen unterschiedliche Werte liefern kann, was zu sehr schwer auffindbaren sporadisch auftretenden Fehlern führen könnte. Siehe dazu auch über "EnableDelayedExpansion" unten ;
2. %time% vor 10 Uhr eine Zeit ohne führende Null liefert. Stattdessen erhält man ein Leerzeichen, was bei einem sortiergerechten Zeitstempel zu Problemen führen kann. So zum Beispiel werden die Datei-/Ordernamen mit dem Zeitstempel einen Leerschritt beinhalten, was eine entsprechende Benutzung von Anführungszeichen anfordert. Die If-Bedingung prüft in der Variablen SORTTIME, ob das erste Zeichen ein Leerzeichen ist. Ist das vor 10 Uhr morgens der Fall, wird das Leerzeichen durch eine 0 ersetzt:

```
set t=%time%
set SORTTIME=%t:~0,2%%t:~3,2%%t:~6,2%
if "%SORTTIME:~0,1%"==" " set SORTTIME=0%SORTTIME:~1,6%
```

Beim Datum ist diese Angabe hier nicht nötig, da dieses standardmäßig im tt.mm.jjjj-Format ausgegeben wird. Wer aber nur das Jahr haben will, kann "%date:~-4%" eingeben und erhält damit die letzten 4 Zeichen. Für ein sortiergerechtes Datum in der Umgebungsvariablen sortdate sorgt z. B.

```
set d=%date%
set SORTDATE=%d:~-4%-%d:~3,2%-%d:~0,2%
echo %SORTDATE%
```

Wert der Umgebungsvariablen: **2009-04-20**

**Beachte:** In einer Batchdatei kann die Verwendung von %DATE% und insbesondere von %TIME% dazu führen, dass sich die ausgegebene Uhrzeit nicht aktualisiert. Hierzu folgendes Beispiel:

```
@echo off
echo ## Die aktuelle Zeit ist: %TIME%
echo ## bitte 5 Sec. warten ...
ping localhost -n 5 > NUL
echo ## jetzt sind genau 5 Sec. vergangen ,TIME liefert %TIME%, das ist noch OK
echo ## doch in der FOR Schleife wird bereits die alte Zeit verwendet.
for /L %%N IN (0, 1, 3) DO (
    echo %time%
    pause
)
echo ## und dies bleibt für jede Ausgabe innerhalb der FOR-Schleife so.
echo.
echo ## Auch in z.B. IF-Anweisungen ist das so.
if TRUE==TRUE (
    echo 1. Zeit in der If Anweisung: %TIME%
    echo Warte 5 Sec.
    ping localhost -n 5 > NUL
    echo 2. Zeit in der If Anweisung: %TIME%
    echo Warte nochmals 5 Sec.
    ping localhost -n 5 > NUL
    echo 3. Zeit in der If Anweisung: %TIME%
)
echo ## Dabei ist es bereits: %TIME%
pause
```

Damit %DATE% und %TIME% die richtigen Werte ausgeben, muss unbedingt die verzögerte Erweiterung von Umgebungsvariablen mit dem SetLocal Befehl<sup>[2]</sup>

```
SetLocal EnableDelayedExpansion
```

aktiviert werden.

Hier das korrekte Beispiel:

```
@echo off
SetLocal EnableDelayedExpansion
echo ## Die aktuelle Zeit ist: %time:~0,8%
echo ## bitte 5 Sec. warten ...
ping localhost -n 5 > NUL
echo ## jetzt sind 5 Sec. vergangen, TIME liefert %TIME%, das ist OK
echo ## Jetzt gibt auch die FOR Schleife die korrekte Zeit aus.
for /L %%N IN (0, 1, 3) DO (
    echo !TIME!
    pause
)
echo.
echo ## Auch in z.B. IF-Anweisungen ist es jetzt richtig.
if TRUE==TRUE (
    echo 1. Zeit in der If Anweisung: !TIME!
    echo Warte 5 Sec.
    ping localhost -n 5 > NUL
    echo 2. Zeit in der If Anweisung: !TIME!
    echo Warte nochmals 5 Sec.
    ping localhost -n 5 > NUL
    echo 3. Zeit in der If Anweisung: !TIME!
)
echo ## Es ist jetzt: %TIME%
EndLocal
pause
```

## Lokalisierungsunabhängige Datumsermittlung

Wenn man Skripte häufig auf unterschiedlichen Umgebungen laufen lassen will, ist eine sprachunabhängige Zeit- und Datumsermittlung erforderlich.

Dies kann wie folgt erreicht werden:

```
for /f %%x in ('wmic path win32_localtime get /format:list ^| findstr "=") do set %%x
REM Fuehrende Nullen ergaenzen falls der Wert einstellig ist
if "%Month:~1,1%"==" " set Month=0%Month%
if "%Day:~1,1%"==" " set Day=0%Day%
if "%Hour:~1,1%"==" " set Hour=0%Hour%
if "%Minute:~1,1%"==" " set Minute=0%Minute%
if "%Second:~1,1%"==" " set Second=0%Second%
set MYDATE=%Year%-Month%-Day%
set MYTIME=%Hour%.%Minute%H
```

## Ausgaben besser anzeigen

Wenn man nicht die Ausgabe von Befehlen per **@echo off** "Ausblendet" kann man am besten das Prompt ändern, sodass man besser erkennen kann, was passiert:

```
@prompt -$G
```

Der Prompt ist dann ->

## Unterroutinen und Unterprogramme

Unterroutinen kann man mittels **goto** oder **call** und Unterprogramme mit Hilfe von **call** realisieren.

```
call:unterroutine Hallo
echo Fertig!
goto:eof

:unterroutine
echo Übergebener Parameter an Unterroutine: %1
goto:eof
```

Beachten Sie, dass Sie beim Aufruf von Unterroutinen per **call** Probleme mit Filehandles bekommen können. Dies liegt darin begründet, dass ein Aufruf per **call** als Aufruf eines Unterprogramms interpretiert wird, während es sich bei **goto** stets um Unterroutines handelt.

*Anmerkung:* **goto:eof** ist eine Spezialmarke mit der Sie stets zum Ende Ihres Skriptes (bzw. In Unterroutinen zurück zum Aufruf) springen.

## Benutzereingaben mittels "set /P"

```
@echo off
set /P w= [i]nstallieren / [d]eInstallieren?
REM die option /I beim if bewirkt, dass nicht
REM zwischen Gross und Kleinschreibung
REM unterschieden wird.
if /I "%w%"=="i" goto Install
if /I "%w%"=="d" goto Deinstall
echo Fehler: [%w%]
goto ende
```

```

:Install
  echo "installieren" ausgewählt
goto ende

:Deinstall
  echo "deinstallieren" ausgewählt
goto ende

:ende
  echo.
  pause

```

## stdout in Umgebungsvariable speichern

Falls man den **stdout** in einer Umgebungsvariablen speichern möchte, muss man das komplizierter umsetzen. Es gibt zwei verschiedene Möglichkeiten dies anzugehen. `befehl | set /P variable=` funktioniert nämlich nicht. Stattdessen braucht man:

```

BEFEHL > temp.txt
set /p BefehlOutput= < temp.txt
del temp.txt

```

Oder:

```

FOR /F %i IN ('BEFEHL') DO set BefehlOutput=%i

```

Oder mit "usebackq"-Option:

```

FOR /F "usebackq" %i IN (`BEFEHL`) DO set BefehlOutput=%i

```

Die Zeichenkette zwischen den einfachen Anführungszeichen wird dabei als Befehlszeile betrachtet und von einer untergeordneten CMD.EXE ausgeführt. `%BefehlOutput%` kann nun beliebig gebraucht werden.

**Beispiel:**

**Code:**

```

@echo off
FOR /F %i IN ('CD') DO set verzeichnis=%i
echo %verzeichnis%

```

**Ausgabe:**

```

C:\Programme\Batch

```

Vorsicht ist geboten bei Befehlen, welche mehrzeilige Ausgaben produzieren und bei solchen, welche in ihrer Ausgabe auch Leerzeichen enthalten können. Da das Standardtrennzeichen ein Blank ist, muss man, wenn man nicht will, dass die Variable nur bis zum Blank gefüllt wird, das Standardtrennzeichen verändern. `FOR /F "delims=" %i IN ('CD') DO set verzeichnis=%i` entfernt jede Art von Trennzeichen. Bei Befehlen, welche mehrzeilige Ausgaben zur Folge haben, bleibt jeweils die letzte Zeile in der Variablen erhalten.

## Dateien und Verzeichnisse auflisten

Hier ist ein Beispiel, in dem alle Dateien, auf welche die Filterbedingung zutrifft, aufgelistet werden. Außerdem werden die Dateianzahl und die Dateigrößen addiert.

```
@echo off
set Filter=*. *
set /A DateiAnzahl=0
set bytes=0

for /R %pfad% %%f in (%Filter%) do (
    set /A DateiAnzahl += 1
    echo %%f - %%~zfBytes
    set /A bytes=bytes+%%~zf
)

echo.
echo %~dp0%Filter%
echo Es sind %DateiAnzahl% Dateien vorhanden.
echo Alle Dateien zusammen: %bytes%Bytes
set /A kbytes=bytes/1024
echo umgerechnet sind das %kbytes% KBytes
echo.
pause
```

## Pause

Oft ist es hilfreich, dass nach dem Beenden des Batch-Programms das Eingabeaufforderungsfenster offen bleibt. So kann man Ausgaben nachlesen oder evtl. aufgetretene Fehler entdecken. Nun könnte man einfach am Ende eine **pause** einfügen. Dabei kann der User einfach das Fenster schließen oder ENTER drücken. Man kann aber auch einfach eine zeitliche Pause mit **ping** realisieren:

```
1 @echo off
2 echo Ich schließe gleich.
3 @ping localhost -n 2 >NUL
```

Dabei kann man die Zeit mit dem Parameter `-n` variieren.

Bei installiertem Windows Server 2003 Resource Kit Tools (<https://www.microsoft.com/en-us/download/details.aspx?id=17657>) steht der Befehl "sleep" zur Verfügung, welcher dieselbe Funktionalität (zeitliche Pause) bietet.

Seit Windows XP<sup>[3]</sup> kann man auch den Befehl **timeout** nutzen:

```
1 @echo off
2 echo Jetzt kommt eine Pause von 20 Sekunden, zum vorzeitigen Ende eine Taste drücken
3 timeout /t 20
4 echo Jetzt musst du 10s warten, Tastendrucke werden ignoriert
5 timeout /t 10 /nobreak
6 echo Und jetzt geht es erst weiter, wenn du eine Taste drückst
7 timeout /t -1
```

Die Anzahl an Sekunden wird mit dem Parameter `/t` übergeben, wird ein negativer Wert angegeben wird unendlich gewartet. Der Befehl beendet sich, wenn man eine Taste drückt, es sei denn man startet mit dem Parameter `/nobreak`

## Minimiert ausführen

Hin und wieder ist es sinnvoll, dass die Batchdatei minimiert ausgeführt wird (z. B. eine Login-Batch-Datei). Es ist möglich, dass man die Batchdatei normal startet und sie sich selber minimiert ausführt. Der Nachteil ist allerdings, dass sich kurzzeitig ein Eingabeaufforderungs-Fenster öffnet.

```
@echo off
if not "%1"==" goto %1

start /MIN cmd.exe /C "%~nx0 begin"
goto:eof

:begin
echo Hallo, ich laufe minimiert!
pause
goto:eof
```

Noch eine Konstruktion ganz ohne Labels, nach diesem Newsgroup-Beitrag (<http://groups.google.co.uk/group/alt.msdos.batch.nt/msg/d11e951a822bc420?dmode=source>)

```
@set !=||(set !=1&start "%~dpx0" /min cmd /c %0 %*&set !=&goto :eof)
```

Anmerkung: Sollte die Command-Processor-Option "DelayedExpansion" in der Registry aktiviert sein (siehe unter Hilfe "cmd /?"), lässt sich ein "!" als Variablenname **nicht** verwenden. In diesem Fall -bzw. sinnvollerweise immer- den Variablennamen ändern auf x oder y oder # oder @....

Beispiel:

```
@set #=||(set #=1&start "%~dpx0" /min cmd /c %0 %*&set #=&goto :eof)
```

Noch eine Variante der oben gezeigten Beispiele, die bei mir funktioniert, da ich mit der Konstruktion 'ohne Labels' auf einem x64 System Probleme hatte:

```
if "%1"==" start /min cmd.exe /C "%~dpx0 x"&goto :eof
```

## Mittels start /LOW die Priorität festlegen

Manchmal ist es hilfreich, wenn die Batchdatei mit einer niedrigen Priorität läuft. Das kann man mittels **start /LOW** erreichen. Weitere Optionen sind **NORMAL**, **HIGH**, **REALTIME**, **ABOVENORMAL** und **BELOWNORMAL**. Das Beispiel zeigt, wie eine Batchdatei quasi sich selber in die niedrige Priorität versetzen kann. In dem Fall klappt es allerdings nur, wenn beim ersten Start kein Parameter übergeben wurde.

```
@echo off
if "%1"==" (
start /WAIT /LOW /B cmd.exe /V /C "%~0" weiter_machen
goto:eof
)
echo Jetzt laufe ich mit niedriger Priorität!
echo Überprüfe es im Taskmanager!
pause
```

Funktionsweise: Das Prinzip ist eigentlich ganz einfach. Wenn kein Parameter übergeben wird, wird angenommen, daß die Batchdatei zum ersten mal gestartet wurde. Die **if "%1"=="** Bedingung ist also erfüllt. Mittels **start** wird dann dieselbe Batchdatei mit veränderter Priorität

gestartet, allerdings mit einem angehängten Parameter **weiter\_machen** (Könnte auch irgendwas anderes sein!). Somit ist beim nächsten Aufruf die **if "%1"=""** Bedingung nicht mehr erfüllt und der normale Teil der Batchdatei wird abgearbeitet.

Wenn man der Batchdatei einen Parameter übergeben möchte (z. B. ein Dateiname o.ä.) muss man alle Parameter verschieben:

```
@echo off
if "%2"==" " (
    start /WAIT /LOW /B cmd.exe /V /C "%~0" %1 weiter_machen
    goto:eof
)
echo Jetzt laufe ich mit niedriger Priorität!
echo Nun kann [%1] 'bearbeitet' werden...
pause
```

## Probleme mit Variablen

Wenn man sich die Hilfeseiten zu **set**, mittels **set /?** durchliest (weitere Informationen unter **setlocal /?** und **cmd /?**), stößt man auf das Thema *verzögerte Erweiterung von Variablen*. Das will ich hier mal anhand von Beispielen erklären:

### Das Problem

```
set test=1
if "%test%"=="1" (
    set test=2
    echo Wert von 'test' im IF-Block: %test%
)
echo Wert von 'test' nach IF-Block: %test%
```

Man sollte meinen, dass der Wert von %test% in beiden Ausgaben **2** ist. Doch *leider* ist es nicht so. Denn innerhalb des IF-Blocks wird das Neusetzen der Variable test von 1 auf 2 noch nicht aktiv und somit ist das Ergebnis **Wert von 'test' im IF-Block: 1** Erst nach dem IF-Block ist der Wert aktualisiert: **Wert von 'test' nach IF-Block: 2**

### Lösung: cmd.exe /V:ON

In einer Batchdatei, die mit **cmd /V:ON** gestartet wurde, werden Variablen innerhalb von Befehlsblöcken aktualisiert. Jedoch kann man sie nicht gewohnt mit **%test%** ansprechen, sondern mit **!test!**. Der Nachfolgende Code startet automatisch die Batch Datei neu mit eingeschalteter *verzögerte Erweiterung von Variablen* und übergibt sämtliche Eingabeparameter an sich selbst.

```
@echo off
set delayedExpansion=off
if "true"=="true" (
    set delayedExpansion=on
    if "!delayedExpansion!" NEQ "on" (
        cmd.exe /V:ON /C "%~f0 %*"
        goto:eof
    ) else (
        REM Der Code kann hier
    )
)
REM sowie hier eingefügt werden.
REM Start des Beispiels
set test=1
if "%test%"=="1" (
    set test=2
```

```

    echo Wert von 'test'-Prozent in dem IF-Block: %test%
    echo Wert von 'test'-Ausrufezeichen in dem IF-Block: !test!
)
echo Wert von 'test' nach dem IF-Block: %test%
pause
REM Ende des Beispiels
@echo on
REM Wegen der doppelten Ausführung muss unbedingt ein 'exit' am Ende stehen.
exit

```

## Lösung: setlocal EnableDelayedExpansion

Mit **setlocal EnableDelayedExpansion** wird die *verzögerte Erweiterung von Variablen* nur bis zum dazugehörigen **endlocal** oder dem Ende der Batch Datei aktiviert.

```

@echo off
setlocal EnableDelayedExpansion
REM Start des Beispiels
set test=1
if "%test%"=="1" (
    set test=2
    echo Wert von 'test'-Prozent in dem IF-Block: %test%
    echo Wert von 'test'-Ausrufezeichen in dem IF-Block: !test!
)
echo Wert von 'test' nach dem IF-Block: %test%
pause
REM Ende des Beispiels
endlocal
@echo on
exit

```

## Ausgaben/Fehler unterdrücken

Manchmal möchte man per Batch ein Programm starten aber es soll dabei keine Ausgabe gemacht werden. Das ist recht einfach:

```
MeinProgramm.exe >NUL
```

Es könnte aber sein, dass evtl. Fehler dennoch ausgegeben werden. Das liegt daran, daß die Programme in dem Fall auf **stderr** statt **stdout** schreiben. Um auch in dem Fall die Ausgabe zu unterdrücken, kann man mit einem zusätzlichen **2>&1** die Ausgaben von stderr auf stdout umleiten. Da stdout dann nach **NUL** verschoben wird, sieht man absolut nichts:

```
MeinProgramm.exe >NUL 2>&1
```

## Professionelle Message-Fenster erzeugen

Bisher konnte man in Batch keine Fenster erzeugen, bzw. nur in Windows XP mithilfe des Windows Nachrichtendienstes. Doch der ist für Anwendungen viel zu unpraktisch, da man nicht einmal den Fenstertitel bestimmen kann und außerdem ist dieser unter Vista oder Windows 2000 nicht verfügbar.

Ich habe jedoch ein kleines Schlupfloch gefunden, mit dem man dennoch Fenster erzeugen kann:

```

@echo off
Echo msgbox "Text",0,"Fenstername" >Test.vbs
ping localhost -n 3 >NUL

```

```
start Test.vbs
pause
```

Erklärung: Man erzeugt hier mithilfe des "Größer als" Symbols (Eng.: greater than) > einen Temporären VBScript, der in der Lage ist, ein Messagefenster zu erzeugen. Dies sorgt in Anwendungen für mehr Übersicht und Professionalität.

## Falls das nicht funktioniert

Falls statt eures Textes die Meldung :

"Der Zugriff auf den Windows Scripthost ist auf diesem Computer deaktiviert"... erscheint, dann ist das Öffnen von VBscripts und JScripts aus Sicherheitsgründen verboten.

Um das zu ändern, öffnet ihr den Windows Registrierungseditor (regedit.exe) und löscht NUR folgenden Wert :

**HKey\_Local\_Machine\Software\Microsoft\WindowsScriptHost\Enabled**

dann dürfte es funktionieren.

## Sicherheitshinweise:

- Erstellt vor der Änderung eine Sicherheitskopie eurer Werte!
- Die "Sicherheitsgründe" verringern unter anderem die Angriffsfläche für Würmer u.ä. Schädlinge, dies sollte bei einer Änderung der Sicherheitsrichtlinien per Registrierungseditor nicht vollständig vernachlässigt werden.

## Status über bearbeitete Zeilen ausgeben

Häufig bearbeitet man eine Liste von Objekten mit einer FOR-Schleife. Damit man darüber informiert ist wie weit die Bearbeitung bereits fortgeschritten ist, kann man die Anzahl der Bearbeiteten Objekte im titel der DOS-Box ausgeben lassen.

Schritt 1 - Ermitteln / zählen der zu bearbeitenden Zeilen: dies geht am besten mit folgender FOR-Schleife:

```
FOR /F "eol=# tokens=1,2,3 " %A IN ('find /c ";"liste.txt') DO echo %C Zeilen
```

Wobei das ";" ein Zeichen sein muss das in jeder zu verarbeitenden Zeile vorkommt. Dies ist am schnellsten. Will man wirklich jede Zeile zählen unabhängig vom Inhalt, dann kann man auch folgenden Befehl verwenden:

```
FOR /F "delims=:" %%A IN ('findstr /N .* "liste.txt"') DO echo %%A Zeilen
```

Jetzt bedarf es noch des *SET*-Befehl zum Berechnen der aktuellen Zeile und *setLocal EnableDelayedExpansion*, dann könnte das Script folgendermaßen aussehen:

```
@echo off
setLocal EnableDelayedExpansion
set COUNT=0
set COUNTMAX=0
```

```
FOR /F "eol=# tokens=1,2,3 " %%A IN ('find /c ";"liste.txt') DO set COUNTMAX=%C

for /f "eol=# tokens=1 delims=;" %%j in (liste.txt) do (
  set /A COUNT +=1
  title %0 - !COUNT! Zeilen von !COUNTMAX! bearbeitet
  echo Tue etwas mit dem Token %%j
  REM der ping wird nur zur Verzögerung ausgeführt damit man der Titel Zeile besser verfolgen kann.
  ping -n 1 localhost >NUL
)
```

## Einzelnachweise

1. SS64.com: *Variables: extract part of a variable (substring)* (<https://ss64.com/nt/syntax-substring.html>) Command line reference
2. Microsoft: *Windows IT Pro Center: Docs / Windows Server / Windows Commands / Commands by Server Role / setlocal* (<https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/setlocal>)
3. [https://technet.microsoft.com/de-de/library/cc754891\(v=ws.10\).aspx](https://technet.microsoft.com/de-de/library/cc754891(v=ws.10).aspx) ([https://technet.microsoft.com/de-de/library/cc754891\(v=ws.10\).aspx](https://technet.microsoft.com/de-de/library/cc754891(v=ws.10).aspx)), Kommandozeilenbefehl Timeout bei Microsoft Technet

# Netzwerkumgebung

## ping

Kann verwendet werden, um die Verbindung zu einem anderen Rechner zu testen. Wenn die Gegenseite antwortet, werden Informationen über die IP-Nummer und zur Verbindungsqualität angezeigt. Zu beachten ist dabei, dass aus einer ausbleibenden Antwort nicht zwangsläufig auf einen missglückten Verbindungsversuch geschlossen werden darf: manche Administratoren unterbinden diese Art Test aus Sicherheitsgründen.

### Beispiel

```
C:\>ping de.wikibooks.org
```

### Ausgabe

```
Ping rr.knams.wikimedia.org [145.97.39.155] mit 32 Bytes Daten:

Antwort von 145.97.39.155: Bytes=32 Zeit=67ms TTL=57
Antwort von 145.97.39.155: Bytes=32 Zeit=35ms TTL=57
Antwort von 145.97.39.155: Bytes=32 Zeit=34ms TTL=57
Antwort von 145.97.39.155: Bytes=32 Zeit=39ms TTL=57

Ping-Statistik für 145.97.39.155:
    Pakete: Gesendet = 4, Empfangen = 4, Verloren = 0 (0% Verlust),
    Ca. Zeitangaben in Millisek.:
    Minimum = 34ms, Maximum = 67ms, Mittelwert = 43ms

C:\>
```

### Hinweis

Dies lässt sich durch einen Trick mit dem find Befehl auch automatisieren:

```
ping de.wikibooks.org | find "TTL" || goto hostnotfound
```

## ipconfig

- Durch den Parameter */help* erhält man kurze Hilfe.
- Der Aufruf des Befehls *ipconfig* ohne Parameter teilt die eigene IP-Adresse mit.
- Durch den Befehl *ipconfig* lässt sich die IP-Adresse von Netzwerkverbindungen (Ethernetkarten) verwalten.

### Beispiel

```
C:\>ipconfig
```

### Ausgabe

```
Windows-IP-Konfiguration
```

```
Ethernetadapter VMware Network Adapter VMnet8:
```

```
Verbindungsspezifisches DNS-Suffix:  
IP-Adresse. . . . . : 192.168.0.1  
Subnetzmaske. . . . . : 255.255.255.0  
Standardgateway . . . . . : 192.168.0.11
```

```
Ethernetadapter VMware Network Adapter VMnet1:
```

```
Verbindungsspezifisches DNS-Suffix:  
IP-Adresse. . . . . : 192.168.2.1  
Subnetzmaske. . . . . : 255.255.255.0  
Standardgateway . . . . . : 192.168.2.11
```

```
Ethernetadapter LAN-Verbindung 3:
```

```
Verbindungsspezifisches DNS-Suffix: myhost.local  
IP-Adresse. . . . . : 192.168.1.1  
Subnetzmaske. . . . . : 255.255.255.0  
Standardgateway . . . . . : 192.168.1.11
```

```
C:\>
```

## ipconfig /all

Eine detailliertere Auskunft erhält man mit dem Parameter */all*

### Beispiel

```
C:\>ipconfig /all
```

### Ausgabe

```
Windows-IP-Konfiguration
```

```
Hostname. . . . . : cestmoi  
Primäres DNS-Suffix . . . . . : mypc.local  
Knotentyp . . . . . : Unbekannt
```

```

IP-Routing aktiviert. . . . . : Nein
WINS-Proxy aktiviert. . . . . : Nein
DNS-Suffixsuchliste . . . . . : mypc.local
                                mypc.local

```

Ethernetadapter VMware Network Adapter VMnet8:

```

Verbindungsspezifisches DNS-Suffix:
Beschreibung. . . . . : VMware Virtual Ethernet Adapter for VMnet8
Physikalische Adresse . . . . . : 01-23-45-67-89-AB
DHCP aktiviert. . . . . : Nein
IP-Adresse. . . . . : 192.168.0.1

```

[ ... ]

## ipconfig /renew

Sollten die Netzwerkkonfigurationen nicht mehr aufzufinden sein, dann hilft oft der Parameter /renew

### Beispiel

```
C:\> ipconfig /renew
```

## ipconfig /flushdns

### Situation

Eine Homepage wird nicht angezeigt, stattdessen wird eine Fehlermeldung angezeigt, wie z.B. *"Die Website könnte vorübergehend nicht erreichbar sein, versuchen Sie es bitte später nochmals."*

Als Test wird der Server angepingt, und es kommt eine Fehlermeldung

```
Zeitüberschreitung der Anforderung.
```

Der Administrator kümmert sich um das Problem und sagt, der Server sei wieder OK, aber noch immer kommen Fehlermeldungen und Homepages können nicht angezeigt werden. Dann kann es daran liegen, dass der DNS-Cache noch die Fehlmeldung gespeichert hat, und dieser muss geleert werden. Anstelle den Rechner neu zu starten gibt man folgendes ein

```

C:\>ipconfig /flushdns

Windows-IP-Konfiguration
Der DNS-Auflösungscache wurde geleert.

C:\>

```

## ipconfig /release

Gibt die vom DHCP Server zugeteilte IP-Adresse zurück. Anschliessend kann mit ipconfig /renew eine neue Adresse vom DHCP-Server geholt werden.

## tracert

Mit `tracert` kann man die Route vom eigenen PC zur Zieladresse verfolgen, und evtl. Rückschlüsse ziehen, ob und warum eine Verbindung hängt.

## Beispiel

```
C:\>tracert de.wikibooks.org
```

## Ausgabe

```
Routenverfolgung zu rr.knams.wikimedia.org [145.97.39.155] über maximal 30 Abschnitte:
 1  <1 ms    <1 ms    <1 ms    123.456.789.1
 2   24 ms   18 ms   19 ms   adslgw01-fra4.rm-com.net [217.173.128.45]
 3  139 ms   35 ms   36 ms   bbcr01-ams.titannetworks.nl [217.173.143.17]
 4   40 ms   *       43 ms   xsr03.asd002a.surf.net [195.69.144.34]
 5   41 ms   34 ms   35 ms   AZ-500.XSR01.Amsterdam1A.surf.net [145.145.80.21]
 6   39 ms   38 ms   37 ms   KNCSW001-router.Customer.surf.net [145.145.18.158]
 7   41 ms   43 ms   37 ms   gi0-24.csw2-knams.wikimedia.org [145.97.32.29]
 8   38 ms   38 ms   36 ms   rr.knams.wikimedia.org [145.97.39.155]
Ablaufverfolgung beendet.
C:\>
```

## netstat

Dieser Befehl zeigt alle offenen TCP/IP/ICMP/UDP (usw...) Verbindungen an.

```
netstat
```

Mit der Erweiterung `-p` Protokoll lassen sich auch nur Verbindungen zum Entsprechenden Protokoll auslesen(meist TCP):

```
netstat -p TCP
```

## netsh

Ohne Parametereingabe ergibt sich ein neuer Kontext. Weitere Hilfe mit: `?`

Beispiel : `netsh interface dump > ipconf1.dat`

Diese Zeile als Batch-Datei ausgeführt legt die momentane IP-Konfiguration in einer Datei namens `ipconf1.dat` ab. Umgekehrt kann aus einer Datei z. B. namens `ipconf2.dat` eine neue IP-Konfiguration geladen und übernommen werden:

```
netsh -f ipconf2.dat
```

So lässt sich eine IP-Konfiguration ohne Umherklicken in Systemmenüs ablegen und (modifiziert) wieder zurückholen.

## NET

NET ist ein Tool zur Überwachung und Manipulation von Netzwerkaktivitäten sowie Windows-Diensten. Es hat einen recht großen Funktionsumfang, von dem ich hier jedoch nur einige kurz vorstelle.

## NET USE - Netzlaufwerke verbinden

NET USE kann genutzt werden um eine Verbindung zu Netzwerk- oder Samba-Freigaben herzustellen. Üblicherweise wird hierbei ein Laufwerksbuchstabe verwendet, dies ist jedoch nicht zwingend.

### Syntax:

```
NET USE x: \\name.oder.ip.des.rechners\freigabe
```

Wenn für die Verbindung die Eingabe von Zugangsdaten erforderlich ist können diese auch mit angegeben werden:

```
NET USE x: /USER:benutzer \\name.oder.ip.des.rechners\freigabe "passwort"
```

Wenn für einen Rechner Zugangsdaten benötigt werden und diese nicht oder falsch angegeben wurden, so wird der Benutzer nach diesen gefragt. Es kann auch nur ein Benutzer angegeben werden indem das Passwort durch einen Asterisk (\*) ersetzt wird.

Die Eingabe von NET USE ohne Parameter zeigt bereits verbundene Laufwerke an. Bei Windows-Freigaben wird auch ein Status angezeigt, bei Samba-Freigaben jedoch nicht.

## NET SEND - Nachrichten an andere Rechner senden

Mit NET SEND können kurze Nachrichten an andere Rechner geschickt werden. Der Empfänger sieht diese dann normalerweise in einer Message-Box. Die Benutzung empfiehlt sich nur bedingt, da moderne Firewalls solche Nachrichten normalerweise blocken (siehe Hinweis unten).

### Syntax:

```
NET SEND Empfänger Nachricht
```

Empfänger kann dabei sein:

- IP-Adresse
- Domain
- Domain mit Benutzerangabe (/domain:name)
- Rechnername im lokalen Netzwerk
- Benutzername im lokalen Netzwerk
- Asterisk für alle erreichbaren (lokalen) Rechner

**Hinweis:** Da NET SEND leider sehr oft für Spam missbraucht wird/wurde empfiehlt es sich es abzuschalten. Dies geschieht über den Windows-Service "Nachrichtendienst". Dadurch können weder Nachrichten gesendet noch empfangen werden. (NET STOP Messenger)

## NET START

Startet Windows-Dienste

### Syntax:

```
NET START Dienstname (Entweder "Messenger" oder "Nachrichtendienst", je nach Regions- / Spracheinstellungen)
```

Die Eingabe von NET START ohne Parameter zeigt die aktuell gestarteten Windows-Dienste an.

## NET STOP

Stoppt Windows-Dienste

Gleiche Syntax wie NET START

## NET VIEW

Zeigt erreichbare Computer des lokalen Netzwerks, bzw. deren Freigaben an.

### Syntax:

```
NET VIEW [\\computername]
```

Einfache Eingabe von NET VIEW zeigt alle Computer an, wenn ein Computername mit angegeben wird werden die Freigaben angezeigt. Dies funktioniert nur bei Windows-Rechnern. UNIX-Rechner mit Samba-Freigaben werden zwar in der Übersicht angezeigt, die Freigaben können aber nicht abgefragt werden.

## NET USER

Mit diesem Befehl können Benutzerkonten verwaltet werden. Dazu sind Administratorrechte erforderlich.

### Syntax:

```
NET USER
```

Dieser Befehl allein zeigt nur alle Benutzer des Rechners an. (2000) (XP)

```
Net User <name>
```

Durch diesen Befehl wird die Einstellung des Benutzerkontos des Benutzers <name> angezeigt. (2000) (XP)

```
Net User <name> <passwort> /add
```

Dieser Befehl fügt einen Benutzer hinzu. (2000) (XP)

```
Net User <name> /delete
```

Und dieser Befehl wiederum löscht einen Benutzer. (2000) (XP)

```
Net User <name> <passwort>
```

Dieser Befehl verändert das Passwort des Benutzers. Wenn anstatt des passworts ein Stern \* angegeben wird, dann wird das neue Passwort abgefragt. (2000) (XP)

```
Net User <name> /expires:never
```

Damit wird eingestellt, dass das Passwort des Benutzers <name> niemals abläuft. (2000) (XP)

```
Net User <name> /active:Yes
Net User <name> /active:No
```

Damit wird das entsprechende Benutzerkonto aktiviert oder deaktiviert. (2000) (XP)

```
Net User <name> /passwordchg:Yes
Net User <name> /passwordchg:No
```

Damit wird eingestellt, ob der Benutzer sein eigenes Passwort verändern darf. (2000) (XP)

```
Net User <name> /comment:"Herr Maier vom Verkauf"
```

Damit wird der Benutzerkommentar verändert. (2000) (XP)

```
Net User <name> /Fullname:"Karl-Otto Maier-Müller"
```

Damit wird der komplette Benutzername verändert. (2000) (XP)

```
Net User <name> /expires:2009-12-31
```

Damit wird eine Zeit vorgegeben, nach der der Benutzer sein Passwort ändern muss. (2000) (XP)

```
Net User <name> /profilepath="D:\Maier"
Net User <name> /scriptpath="D:\Maier"
Net User <name> /homedir="D:\Maier"
Net User <name> /homedirreq=Yes
Net User <name> /homedirreq=No
```

Damit wird eingestellt, wo sich die Benutzerdaten befinden. Der Ort der Benutzerdaten wird in der Umgebungsvariable *USERPROFILE* gespeichert. (2000) (XP)

```
Net User <name> /countrycode=049
```

Damit wird die Ländereinstellung verändert. (2000) (XP)

```
Net User <name> /workstations:*
```

Damit wird es dem Benutzer erlaubt, sich von anderen Rechnern aus an diesem Rechner anzumelden. Anstatt \* kann hier auch eine Liste von Computernamen angegeben werden. Dann darf sich der Benutzer nur von diesen Computern anmelden. (2000) (XP)

Hinweis: Diese Arbeitsvorgänge gehen auch mit der Windows Benutzerkontensteuerung. Diese kann erreicht werden durch [Start] [Einstellungen] [Systemsteuerung] [Benutzer]

Hinweis: Der Name des aktuell angemeldeten Benutzers ist in der Umgebungsvariable `UserName`.

Hinweis: Hilfe zum Befehl `Net User` erhält man durch Eingabe von `Net User /help`

Beispiel: Um den Benutzer *Otto* mit dem Passwort *Frieda* einzurichten, kann folgende Batch-Datei ausgeführt werden:

```
net user Otto Frieda /add
net user Otto /expires:never
net user Otto /comment:"Default USER from This Computer"
net user Otto /active:Yes
net user Otto
```

## Beispiele

- 1.
2. Nummerierter Listeneintrag

### Regedit.exe

Eine kleine Batchdatei, um mit `regedit.exe` direkt einen bestimmten REG-Zweig anzuspringen:

```
set LastKey="HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Hotfix"
set Key=HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Applets\Regedit
reg add %Key% /f /v "LastKey" /d %LastKey%
start regedit.exe
```

Erläuterung:

- **LastKey** ist der Reg-Schlüssel, der als Letztes angezeigt wurde und dieser wird beim Start von `regedit` direkt angesprungen, diesen Schlüssel kann man nach Belieben anpassen.
- **Key** ist der Reg-Schlüssel in dem sich der Wert "**LastKey**" befindet.

**LastKey** wird mittels **reg add** in die Registry eingespielt und danach wird `regedit` gestartet. Das Ganze funktioniert allerdings nur, wenn `regedit.exe` noch nicht läuft.

Und hier noch eine kleine Batchdatei, um mit `regedit.exe` Werte aus der Registry auszulesen:

```
set Key="HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Applets\Regedit"
regedit.exe /e c:\temp\inhalt.reg %Key%
```

Achtung! Die Datei `inhalt.reg` ist im Unicode-Format. Ein Tipp dazu: Der `type`-Befehl kann Unicode lesen.

### TaskList-Abfrage

Eine Routine um das Laufen eines Prozesses mittels **tasklist.exe** festzustellen. In diesem Beispiel wird geprüft, ob **Prozess.exe** läuft.

```
@echo off
set ProgEXE=Prozess.exe
set tempdatei=%temp%\TempDatei.txt

tasklist /FI "IMAGENAME eq %ProgEXE%" /FO CSV>%tempdatei%

for /F "skip=2 tokens=1 delims=," %%f in (%tempdatei%) do (
    echo gefiltert: [%%f]
    if "%%f"=="%ProgEXE%" echo Prozess %ProgEXE% läuft
    goto ende
)
echo Prozess %ProgEXE% läuft nicht!
goto ende

:ende
del %tempdatei%
```

Alternativ, ohne obige temporäre Datei 'tempdatei' zu benötigen, kann man beispielsweise unter Windows XP folgendes verwenden:

```
TaskList /FI "IMAGENAME eq %ProgEXE%" 2>NUL | Find "%ProgEXE%" >NUL
IF %ERRORLEVEL% == 1 (
    ECHO NOT Running.
    GOTO Ende
) ELSE (
    ECHO Running.
    GOTO Ende
)
```

## Dienste einrichten

```
@echo off
prompt -$G
echo.
echo ### Manuelle Dienste ###
echo.
call:M helpsvc      "Hilfe und Support"
call:M ImapiService "IMAPI-CD-Brenn-COM-Dienste"

echo.
echo ### nicht benötigte Dienste Deaktivieren ###
echo.
call:D cisvc        "Indextdienst"
call:D ERSvc        "Fehlerberichterstattungsdienst"

echo.
echo ### Automatische Dienste ###
echo.
call:A CryptSvc     "Kryptografiedienste"
call:A MSIServer    "Windows Installer"

echo.
echo.
echo Fertig!
echo.
pause
goto:eof

:A
echo %2 auf Automatisch
```

```

set Parameter=auto
net start %1
goto machen

:M
echo %2 auf Manuell
set Parameter=demand
net stop %1
goto machen

:D
echo %2 auf Deaktiviert
set Parameter=disabled
net stop %1
goto machen

rem Systemstart      Gerätetreiber, der vom Startladeprogramm geladen wird.
rem system          Gerätetreiber start während der Kernelinitialisierung.
rem auto            automatischer start, nach systemstart, vor Benutzeranmeldung.
rem demand         Dienst, der manuell gestartet werden muss.
rem disabled       Dienst deaktivieren

:machen
  echo on
  sc config %1 start= %Parameter%
  @echo off
  echo.
  echo.
goto:eof

```

## Windows XP SP2: installierte Patches/Updates auflisten

Zeigt unter Windows XP mit Service Pack 2 alle zusätzlich installierten Patches/Updates an.

```

@echo off

set RegHotfixListe=HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Hotfix

set RegHotfixInfo=HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Updates\Windows XP\SP3

for /F "delims=\ tokens=7" %%f in ('reg query "%RegHotfixListe%"') do (
  set HotFixNr=%%f
  CALL :Anzeigen
)

echo.
echo.
pause
goto:eof

:Anzeigen
reg query "%RegHotfixListe%\%HotFixNr%" /v Installed | FIND "0x1">NUL
if "%errorlevel%"=="0" (
  set installiert=ja
  call:KB_Info
) ELSE (
  set installiert=nein
)

echo %HotFixNr% - Ist Installiert: %installiert%
if not "%InstalledBy%"==" " echo %HotFixNr% installiert von %InstalledBy%
if not "%InstalledDate%"==" " echo %HotFixNr% am %InstalledDate%
echo.
goto:eof

```

```

:KB_Info
  set InstalledBy=
  for /F "tokens=3" %%c in ('reg query "%RegHotfixInfo%\%HotFixNr%" /v InstalledBy 2^>NUL ^| FIND "InstalledBy"') do
    set InstalledBy=%%c
  )
  set InstalledDate=
  for /F "tokens=3" %%c in ('reg query "%RegHotfixInfo%\%HotFixNr%" /v InstalledDate 2^>NUL ^| FIND "InstalledDate"') do
    set InstalledDate=%%c
  )
goto:eof

```

## Printdateien direkt an den Drucker senden

Sendet einzelne Druckdateien an im Netzwerk freigegebene Drucker. Eignet sich besonders bei Standard Drucksprachen wie Postscript und HPGL/2.

```

@echo off
rem "pl.bat" sendet einzelne Druckdateien an bel. Drucker direkt.
rem eignet sich besonders bei Standard Drucksprachen wie Postscript und HPGL/2.
rem Laesst sich in Windows bequem einbinden, z.B rechte Maustaste - Senden an PLOTter

:voreingestellt
  set server=192.168.0.5
  rem server --- IP von Printserver/oder Name von Freigaberechner ---
  set druckerbeschreibung=192.168.0.5 Freigabename EPSON
  rem druckerbeschreibung --- damit der Drucker leichter identifiziert werden kann ---
  set drucker=EPSONNT
  rem drucker --- das muss der Freigabename im Netzwerk/auf dem Freigaberechner sein ---

  if '%1' == 'drucker01' goto drucker01
  if '%1' == 'drucker02' goto drucker02
  if '%1' == 'canon' goto canon
goto datei

:drucker01
  shift
goto datei

:drucker02
  set server=42.1.1.240
  set druckerbeschreibung=HP DJ 755CM im Sekretariat
  set drucker=hpdj755-tcp
  shift
goto datei

:canon
  set server=192.168.0.1
  set druckerbeschreibung=Canon s/w Laser im EG
  set drucker=CanonLBP
  shift
goto datei

:datei
  if '%~1' == '' goto aufruf
goto plot

:plot
  echo.
  echo Dateiname : %1
  echo Druckserver : %server%
  echo Freigabename: %drucker%
  echo Beschreibung: %druckerbeschreibung%
  echo.
  print /d:\\%server%\%drucker% '%~1'
goto:eof

:aufruf
  echo.
  echo aufruf : pl [plotter] Datei

```

```
echo plotter : drucker01, drucker02, canon, (voreingestellt ist drucker01)
echo.
```

### Anmerkung zur Nutzung dieses Scriptes:

- Der Printbefehl schickt die Datei OHNE Interpretation direkt an den Drucker. Eine Bilddatei im JPEG-Format würde statt der Grafik also lediglich einen haufen wirrer Zeichen ergeben.
- Der Drucker muss in der Lage sein, selbst ASCII-Zeichen zu verarbeiten und auszugeben. Dies ist bei älteren Druckern (vor ca. 1996 sicher) immer der Fall. Heute beherrschen meist nur noch PostScript-fähige Drucker diese Fähigkeit. "Billigen" GDI-Druckern muss man etwas auf die Sprünge helfen mit dem verwendeten "Druckprozessor" im Windows-Druckertreiber.
- Zum Umschalten zunächst die Druckerübersicht öffnen - Zu finden unter Start|Drucker und Faxgeräte
- Dann die Eigenschaften des gewünschten Druckers aufrufen (Rechtsklick|Eigenschaften)
- Auf dem Reiter "Erweitert" den Button "Druckprozessor..." klicken
- In dem erscheinenden Dialogfeld gibt es 2 Felder mit Einträgen. Im rechten Feld (Standarddatentypen) findet sich neben verschiedenen anderen auch der Eintrag "Text". Diesen markieren und dann alles mit OK wieder schließen.

Der Druckprozessor von Windows ist nun in der Lage, Ausgaben entsprechend zu Interpretieren und dem GDI-Drucker als Grafik zu liefern. Es können aber nur stur 80 Zeichen (maximal) auf ca. 60 Zeilen verteilt werden. Manchmal sind es bis zu 66 Zeilen.

## Copy

Kopieren von Dateien.

Der Befehl

```
copy test.txt c:\temp
```

kopiert die Datei *test.txt* aus dem aktuellen Ordner in den Ordner *c:\temp*.

Der Befehl

```
copy c:\test.txt c:\temp
```

kopiert die Datei *test.txt* aus dem Ordner *c:\* in den Ordner *c:\temp*

Datei- und Pfadnamen mit Sonderzeichen ( hier das Leerzeichen sowie `[ ] { } ^ = ; ! ' + , ` ~` ) müssen beim Copy=Kommando und den meisten anderen Kommandos in doppelten Anführungszeichen (") angegeben werden, zum Beispiel:

```
copy f:\Beispiel.cmd "c:\Dokumente und Einstellungen\Siegfried\Eigene Dateien"
```

Wie man sieht, kann man beide Schreibweisen mischen. Dagegen können sich mehrfach vorhandene Anführungszeichen wieder aufheben. Deshalb müssen vom Benutzer angegebene Anführungszeichen erst entfernt werden, bevor man in der eigenen Batchdatei neue setzt:

```
@echo off
echo Kopiert eine Datei in das eigene Benutzerprofil.
rem Die Anführungszeichen in der folgenden Zeile werden nur benötigt,
rem damit zwischen dem Doppelpunkt und der Eingabe ein Abstand entsteht.
set /p name="Bitte geben Sie einen Dateinamen an: "
rem In der folgenden Zeile werden alle Anführungszeichen entfernt, da
rem nach dem Gleichheitszeichen kein Ersattext folgt, siehe auch set/?.
```

```

rem Das Set-Kommando kommt sowohl beim Variablennamen als auch beim
rem Inhalt ohne Anführungszeichen aus.
set name=%name:="-%
rem An dieser Stelle sollte erst geprüft werden, ob die Datei überhaupt
rem existiert, aber das würde den Rahmen hier sprengen.
copy "%name%" "%HOMEDRIVE%\HOMEPATH%\Eigene Dateien"

```

Theoretisch könnte man auch mit den kurzen Dateinamen (8+3) arbeiten, aber diese sind nicht eindeutig und können von Laufwerk zu Laufwerk unterschiedlich sein. Daher besser nicht verwenden!

Kombiniert in der gegebenen Reihenfolge zwei oder mehrere durch "+" verbundene Textdateien in einer neuen Gesamtdatei.

In diesem Zusammenhang muss auf die Bedeutung der Parameter "/A" (für ASCII-Format, der Standardwert) und "/B" (für Binär-Format ) eingegangen werden, die jedem Dateinamen in obigem Beispiel mit einem Leerzeichen Abstand vorangehen können. DOS-Textdateien enthalten ein Dateiendezeichen (Dezimal 26, Hexadezimal 1A). Beim Kopiervorgang darf in der Gesamtdatei nur als letztes Zeichen ein Dateiendezeichen erscheinen, sonst würde der Text nach der ersten Endmarkierung nicht mehr angezeigt. "copy" kopiert ohne das abschließende Zeichen und fügt als letztes wieder eines zur Gesamtdatei hinzu.

Möchte man Dateien mit binären Inhalten zusammenfügen (z.B. PostScript-Dateien, die auch binäre Abschnitte enthalten können), so würde ohne Parameter "/B" ebenfalls nur bis zu einem zufällig vorkommenden Dateiendezeichen kopiert und damit das Ergebnis unbrauchbar.

```

copy /b "seite 1.ps" + /b "seite 2.ps" "Neue Datei.ps"

```

Als Zieldatei sollte man keine der Quelldateien verwenden, da möglicherweise sonst der Inhalt überschrieben wird.

## Message-Ping

Mit Hilfe des folgenden Scripts kann man Windows um eine nützliche Funktionalität erweitern. Mit Hilfe von **mping** erhält man eine Nachricht, sobald der gepingte Computer wieder erreichbar ist.

Um das Script global zu verwenden, muss es z.B. als "mping.cmd" in "**Windows-Installationsordner**\System32" gespeichert werden und kann von nun an z.B. mit "mping google.com" aufgerufen werden.

```

@echo off

:: wenn kein remote Host angegeben ist, dann gehe zu :Syntax
IF [%1]==[] GOTO Syntax
IF [%1]==[/?] GOTO Syntax
IF [%1]==[/h] GOTO Syntax
IF [%1]==[/help] GOTO Syntax

:mainloop
:: pinge den Host 1 mal und leite die Ausgabe nach 'nul' um (keine sichtbare Ausgabe)
ping %1 -n 1 >nul

:: speichere den ErrorLevel in die Variable 'status'
set status=%errorlevel%

:: sende eine Nachricht sobald der remote Host online ist
if %status%==0 (msg %username% %1 online && exit 0)

:: minimalistische Fortschrittsanzeige

```

```
(Set /P i=.) < NUL

:: sleep workaround über eine hypothetische Auswahl, die nach 30 Sekunden automatisch beantwortet wird.
:: Wer vorher X drückt, kann den Ablauf auch beschleunigen.
:: Die Variante ist meiner Meinung nach besser als 'ping localhost -n 10 >nul'
:: ab windows 7 aufwärts geht auch 'TIMEOUT /T 10 /NOBREAK > nul'
:: (timeout sekunden /NOBREAK verhindert abbrechen durch tastendruck)
choice /T 30 /N /C X /D X > NUL

:: gehe zurück zum anfang
goto mainloop

:Syntax
echo.
echo Syntax: %~n0 [Hostname oder IP]
echo.
echo mping
echo.
echo Pingt einen offline Host und gibt eine Meldung
echo aus, sobald er wieder online ist
echo.
```

## Defrag mit Endlos-Schleife

Das eingebaute Defrag-Tool erlaubt ja immer nur die gleichzeitige Defragmentierung eines Laufwerks. Außerdem sind oftmals mehrere Durchgänge nötig, bis das Laufwerk wirklich vollständig defragmentiert ist. Mit dieser kleinen Batch Datei werden alle Partitionen immer wieder hintereinander defragmentiert. Wenn man wieder arbeiten will, kann man einfach das Eingabefenster schließen. Auch bei laufendem defrag passiert dabei nichts, weil defrag im Hintergrund die aktuelle Arbeit sauber beendet.

Damit aber nicht defekte Laufwerke einfach blind defragmentiert werden, habe ich eine etwas aufwendige Prozedur eingebaut, die die Laufwerke mit chkdsk erst überprüft. Wenn dabei ein Fehler auftritt, wird mit fsutil das dirty Bit gesetzt. Das bewirkt, dass beim nächsten Systemstart die Fehler behoben werden. Außerdem werden die Partitionen in dem loop ausgelassen, welche schon als dirty markiert sind.

Eine kleine Routine war auch nötig, damit Netzwerklaufwerke ausgelassen werden.

```
@echo off
cd /d C:\

:loop
for %i in (c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z) do (
    if exist %i:\NUL call:defrag %i
)
echo.
echo pause...
:: Vorherige Variante 'ping localhost -n 10 >NUL'
choice /t 30 /n /c x /d x >NUL
goto loop

:defrag
call:check %1
if "%ok%"=="nein" (
    echo Laufwerk %1: wird ausgelassen
    echo.
    goto:eof
)
echo.
echo.
echo chkdsk %1: /v
chkdsk %1:
if %errorlevel% NEQ 0 (
    if %errorlevel% NEQ 2 (
        echo %1: chkdsk Fehler!!!
        echo Errorlevel: %errorlevel%
        echo.
```

```

echo setzte dirty-Bit:
fsutil dirty set %1:
echo.
echo Versuche Reparatur
chkdsk %1: /v /x
if %errorlevel% EQU 2 (
    echo Fehler bereinigt!
) else (
    echo Errorlevel: %errorlevel%
    echo.
    goto:eof
)
)
)
echo defrag %1: -f
defrag %1: -f
if %errorlevel% NEQ 0 (
    pause
    exit
)
rem pause...
ping localhost -n 1 >NUL
goto:eof

:check
set ok=nein
net use | find /i "%1:" >NUL
if not errorlevel 1 (
    echo %1: ist ein Netzlaufwerk
    goto:eof
)
fsutil dirty query %1:
for /F "tokens=5" %f in ('fsutil dirty query %1:') do (
    if not "%f"=="NICHT" (
        rem Laufwerk ist als fehlerhaft markiert
        goto:eof
    )
)
)
set ok=ja
goto:eof

```

Als Alternative gibt es das Open Source Program JkDefrag (<http://www.kessels.com/JkDefrag/>), welches auch alle Laufwerke nacheinander defragmentiert.

## Dateiinformatioenen anzeigen

Dieses Batch-Skript zeigt Informationen über die Datei, die als Parameter übergeben wird, an. (beispielsweise per Drag&Drop auf die Batch-Datei)

```

@echo off
echo Attribute.....: %~a1
echo Laufwerk.....: %~d1
echo kompletter Pfad.....: %~f1
echo Dateiname.....: %~n1
echo Ordnerpfad.....: %~p1
echo kompletter Pfad (kurze Form): %~s1
echo Aenderungsdatum.....: %~t1
echo Dateierweiterung.....: %~x1
echo Dateigroesse.....: %~z1
pause

```

Hinweis: Falls die Attribute des eigenen Batch-Skripts ausgelesen werden sollen, so muss der Parameter Null verwendet werden (Beispielsweise %~p0). Eine Kombination ist auch möglich, z.B: %~dp0 (Laufwerk + Pfad)

## Alle verfügbaren PCs im aktuellen Netzwerk suchen

Dieses Batch-Skript pingt alle IP-Adressen eines IP-Bereichs an und zeigt dann die PCs an, die verfügbar sind.

```
@echo off

if exist ips.txt del ips.txt > nul

echo Bitte geben Sie den ersten Teil der IP ein:
set /p ip=

for /L %%N IN (1, 1, 255) DO (
    echo Die IP: %ip%%N wird nun angepingt
    ping %ip%%N -n 1 -w 1 | find "TTL" && echo %ip%%N >> ips.txt
)

cls
type ips.txt
pause >nul
```

Hinweis: Die Datei ips.txt wird nicht gelöscht, damit man sie für weitere Zwecke benutzen kann.

## Hosts Datei mit einem Aliasnamen und der aktuellen IP-Adresse aktualisieren

Häufig werden Webserver oder ähnliche Programme unter Angabe eines Aliasnamens installiert, damit sie leicht auf einen anderen Rechner verschoben werden können, falls der ursprüngliche Rechner defekt ist. Oder die Applikation läuft unter VM-Ware auf einen Laptop bei dem sich die IP-Adressen häufig ändern. Dann kann mit dem Skript der Hostnamenalias mit der aktuellen IP-Adresse aktualisiert werden. Idealerweise wird das Script automatisch beim Start von Windows oder beim Anmelden des Anwenders ausgeführt.

```
@echo off
set HOST_ALIAS=myserver myserver.mydomain.org
set HOST_FILE=%SystemRoot%\System32\drivers\etc\hosts
set TEMP_FILE=%TEMP%\hosts.tmp

REM vorhandenen Eintrag entfernen und den Rest in TEMP-Datei schreiben
type %HOST_FILE% | find /v "myserver" > %TEMP_FILE%

REM Neuen Eintrag wieder hinzufügen
for /F "tokens=15" %i in ('ipconfig ^| find "IP Ad"') do set IP=%i
echo %IP% %HOST_ALIAS%>>%TEMP_FILE%
echo ## Added %IP% %HOST_ALIAS% to the hostsfile

REM Temp-File nach %SystemRoot%\System32\drivers\etc\hosts kopieren
REM Die Datei Hosts darf nicht mit "move" verschoben werde, sondern muss kopiert werden, da sonst der Hostname nicht
copy /y %TEMP_FILE% %HOST_FILE%
del /F /Q %TEMP_FILE%
```

## Windowsversion herausfinden

Mit dieser Batchdatei findet man heraus, welche Windowsversion verwendet wird. Das ist nützlich, wenn man mit bestimmten Pfaden arbeitet, die je nach Betriebssystem verschieden sind. Quelle (<https://www.administrator.de/contentid/56044>)

```
@echo off

rem diese Batch prueft das OS

VER |find /i "Windows 95" >NUL
```

```

IF NOT ERRORLEVEL 1 GOTO 95
VER |find /i "Windows 98" >NUL
IF NOT ERRORLEVEL 1 GOTO 98
VER |find /i "Windows Millennium" >NUL
IF NOT ERRORLEVEL 1 GOTO ME
VER | find "XP" > nul
IF %errorlevel% EQU 0 GOTO XP
VER | find "2000" > nul
IF %errorlevel% EQU 0 GOTO 2000
VER | find "NT" > nul
IF %errorlevel% EQU 0 GOTO NT
VER | find "Microsoft Windows [Version 6.0" > nul
IF %errorlevel% EQU 0 GOTO Vista
VER | find "Microsoft Windows [Version 6.1" > nul
IF %errorlevel% EQU 0 GOTO W7
VER | find "Microsoft Windows [Version 6.2" > nul
IF %errorlevel% EQU 0 GOTO W8
VER | find "Microsoft Windows [Version 6.3" > nul
IF %errorlevel% EQU 0 GOTO W81
VER | find "Microsoft Windows [Version 5" > nul
IF %errorlevel% EQU 0 GOTO 2003
goto unknown
goto end

:unknown
echo unknown
goto end

:95
echo 95
goto end

:98
echo 98
goto end

:ME
echo ME
goto end

:NT
echo NT
goto end

:2000
echo 2000
goto end

:XP
echo XP
goto end

:Vista
echo Vista
goto end

:W7
echo W7
goto end

:W8
echo W8
goto end

:W81
echo Windows 8.1
goto end

:2003
echo 2003
goto end

:end

```

Alternative:

```

@ECHO OFF
FOR /F "tokens=2 delims=[]" %%a IN ('VER') DO FOR /F "tokens=2" %%b IN ("%%a") DO SET VersionNumber=%%b

FOR /F "tokens=1 delims=." %%a IN ("%VersionNumber%") DO SET VersionMajor=%%a
FOR /F "tokens=2 delims=." %%a IN ("%VersionNumber%") DO SET VersionMinor=%%a
FOR /F "tokens=3 delims=." %%a IN ("%VersionNumber%") DO SET VersionBuild=%%a

IF %VersionMajor%==6 (
  IF %VersionMinor%==4 (
    SET VersionName=Windows 10
  ) ELSE (
    IF %VersionMinor%==3 (
      SET VersionName=Windows 8.1
    ) ELSE (
      IF %VersionMinor%==2 (
        SET VersionName=Windows 8
      ) ELSE (
        IF %VersionMinor%==1 (
          SET VersionName=Windows 7
        ) ELSE (
          IF %VersionMinor%==0 (
            SET VersionName=Windows Vista
          ))))
) ELSE (
IF %VersionMajor%==5 (
  IF %VersionMinor%==2 (
    SET VersionName=Windows Server 2003
  ) ELSE (
    IF %VersionMinor%==1 (
      SET VersionName=Windows XP
    ) ELSE (
      IF %VersionMinor%==0 (
        SET VersionName=Windows 2000
      )))
) ELSE (
IF %VersionMajor%==4 ( ECHO 4
  IF %VersionMinor%==90 (
    SET VersionName=Windows ME
  ) ELSE (
    IF %VersionMinor%==10 (
      SET VersionName=Windows 98
    ) ELSE (
      IF %VersionBuild%==1381 (
        SET VersionName=Windows NT 4.0
      ) ELSE (
        IF %VersionMinor%==00 (
          SET VersionName=Windows 95
        )))
) ELSE (
IF %VersionMajor%==3 (
  SET VersionName=Windows 3.1
)))))

ECHO %VersionNumber%
ECHO %VersionMajor%.%VersionMinor%.%VersionBuild%
ECHO %VersionName%
PAUSE

```

## TEMP-Verzeichnis löschen

Im %TEMP%-Verzeichnis sammeln sich mit der Zeit viele Dateien und Unterverzeichnisse an. Mit diesem Skript werden alle auf einmal gelöscht:

```

@echo off

call:clean %TEMP%
IF NOT "%TEMP%" == "%TMP%" (
  call:clean %TMP%
)

pause
goto:eof

```

```

:clean
del /q "%~1\*.*)"
FOR /D %%D IN ("%~1\*") DO (
    rmdir /s /q "%D"
)
goto:eof

```

(Wahrscheinlich erhält man ein paar Fehler, wenn z.B. Dateien geöffnet sind)

## Neue Version

Eine Version für Win XP/2000/WinVista/Win7

```

Echo off
del /f /s /q "%HOMEPATH%\Lokale Einstellungen\Temp\*.*)"
del /f /s /q "%HOMEPATH%\Lokale Einstellungen\Temporary Internet Files\*.*)"
del /f /s /q "%HOMEPATH%\Lokale Einstellungen\Verlauf\*.*)"
del /f /s /q "%windir%\Temp\*.*)"
del /f /s /q "%windir%\Prefetch\*.*)"
del /f /s /q "%windir%\Temp\*.*)"
rmdir /s /q "%HOMEPATH%\Lokale Einstellungen\Temp\"
rmdir /s /q "%windir%\Prefetch\"

diskperf -n
ipconfig /flushdns

```

## FLV/MP4/WAV zu MP3 konvertieren (FFMPEG + Batch)

FFMPEG.EXE nach C:\ kopieren und Batch Datei in einem beliebigen Ordner mit FLV/MP4/WAV Dateien anlegen. Sobald die Batch Datei gestartet wird, geht sie alle FLV/MP4/WAV Files durch und encodiert diese als MP3 (VBR ~226 kbit). Am Ende werden alle MP3 Dateien in den Ordner "MP3" verschoben.

```

mkdir "MP3"
FOR %%X IN (*.flv) DO C:\ffmpeg.exe -y -i "%%X" -q:a 0 "%~nX.mp3"
FOR %%X IN (*.mp4) DO C:\ffmpeg.exe -y -i "%%X" -q:a 0 "%~nX.mp3"
FOR %%X IN (*.wav) DO C:\ffmpeg.exe -y -i "%%X" -q:a 0 "%~nX.mp3"
FOR %%X IN (*.mp3) DO MOVE "%%X" "MP3"

```

Vorteile: Stabil und leistungsstark (FFMPEG + Konsole), keine überladene GUI, man muss keine Bloatware installieren, nutzt Mehrkernprozessoren aus, variable Bitrate, kommt mit Sonderzeichen klar (z.B. "ı"), kopiert nicht nur den Audiostream und Video nach "null" sondern encodiert alles neu (keine "defekten" MP3s mehr bzw. keine "Sprünge" oder sonstiges mehr)

Einfach den Text in eine Textdatei kopieren und zur .bat machen.

## Zusatz-Tools

Viele Aufgaben sind mit der Batchprogrammierung bzw. mit Windows-Scripting alleine kaum oder nur sehr umständlich zu lösen. So kann das Umwandeln von Dateien (Textersetzungen) aufgrund der nicht ganz sauberen Behandlung von Sonder- bzw. Operationszeichen zur Qual werden. Erstaunlich für ein Betriebssystem, welches seit nun mehr über *10 Jahren* - mit Windows 95 - diese Restriktionen offiziell beseitigt hat.

Oft können dafür OpenSource-Tools wie SFK ("Swiss File Knife") (<http://swissfileknife.sourceforge.net/>) und Sed sehr erfolgreich in die Batch-Programmierung eingebunden werden. Für das Verpacken und Komprimieren von Dateien lässt sich das freie 7-Zip (<http://www.7-zip.org/>) gut über die Kommandozeile steuern. Das GPL-Programm Gnuplot eignet sich hervorragend für die Visualisierung von Daten im CSV-Format und lässt sich einfacher und schneller scripten als MS Excel oder OO.org Calc. Zum Herunterladen von Dateien und Webseiten kann Wget verwendet werden. Auch für Batch-Dateien gibt es diverse kostenlose Batch-Compiler, die die Datei in eine ausführbare .exe-Datei umwandeln und den Code vor Manipulation und/oder Copyright-Verletzungen schützen. Für gehobeneren Scripting-Ansprüche und komplexere Vorhaben mit einer längeren Lebensdauer sollten dezidierte Scripting-Sprachen z.B. Awk, Perl, Python, Ruby oder Tcl in Erwägung gezogen werden, für die glücklicherweise das GNU-Projekt *freie* und quelloffene Alternativen bietet.

Eine kostenlose Alternative zum Batching bietet Microsoft selbst auf seiner Homepage mit dem Tool *PowerShell*.

**Weitere Tools sind im Abschnitt "WebLinks" aufgelistet!**

## Referenz

### help

---

Die Hilfe-Funktion steht in allen Windows-Systemen (außer Win9X, d. h. 95/98/ME) zur Verfügung. Der allgemeine Befehl hierzu lautet `help`. Als Ausgabe wird eine Liste mit einem Teil der verfügbaren Kommandozeilen-Befehle ausgegeben. Diese Listen umfasst den Befehlsnamen (linke Spalte) und eine Beschreibung (rechte Spalte). Soll ein Befehl genauer erklärt werden, so lautet die Eingabe `help BEFEHLSNAME` oder auch `BEFEHLSNAME /?`. Diese Eingabe in die Kommandozeile fördert detaillierte Informationen und z.T. auch Beispiele zu einem Befehl zu Tage. Es werden jedoch nicht alle Befehle die es im DOS gibt angezeigt.

### help als Einzelbefehl

Der Befehl `help` listet diverse interne und externe Befehle auf. Diese Liste ist aber bei weitem nicht vollständig, zumal die externen Batch-Befehle durch das Hinzufügen von neuen Programmen ergänzt werden können, z.B. durch die Verwendung von Datenkompressionsprogrammen, die per Kommandozeile aufrufbar sind.

Folgende Befehlsliste erhält man als Ausgabe (in diesem Beispiel unter Windows XP):

<b>ASSOC</b>	Zeigt Dateierweiterungszuordnungen an bzw. ändert sie.
<b>AT</b>	Legt eine Zeit fest, zu der Befehle und Programme auf diesem Computer ausgeführt werden.
<b>ATTRIB</b>	Zeigt Dateiattribute an bzw. ändert sie.
<b>BREAK</b>	Schaltet die erweiterte Überprüfung für STRG+C ein bzw. aus.
<b>CACLS</b>	Zeigt Datei-ACLs (Access Control List) an bzw. ändert sie.
<b>CALL</b>	Ruft eine Batchdatei aus einer anderen Batchdatei heraus auf.
<b>CD</b>	Zeigt den Namen des aktuellen Verzeichnisses an bzw. ändert diesen.

<b>CHCP</b>	Zeigt die aktive Codepagenummer an bzw. legt diese fest.
<b>CHDIR</b>	Zeigt den Namen des aktuellen Verzeichnisses an bzw. ändert diesen.
<b>CHKDSK</b>	Überprüft einen Datenträger und zeigt einen Statusbericht an.
<b>CHKNTFS</b>	Zeigt die Überprüfung des Datenträgers beim Start an bzw. verändert sie.
<b>CLS</b>	Löscht den Bildschirminhalt.
<b>CMD</b>	Startet eine neue Instanz des Windows-Befehlsinterpreters.
<b>COLOR</b>	Legt die Hintergrund- und Vordergrundfarben für die Konsole fest.
<b>COMP</b>	Vergleicht den Inhalt zweier Dateien oder Sätze von Dateien.
<b>COMPACT</b>	Zeigt die Komprimierung von Dateien auf NTFS-Partitionen an bzw. ändert diese.
<b>CONVERT</b>	Konvertiert FAT-Volumes in NTFS. Das aktuelle Laufwerk kann nicht konvertiert werden.
<b>COPY</b>	Kopiert eine oder mehrere Dateien an eine andere Stelle.
<b>DATE</b>	Zeigt das Datum an bzw. legt dieses fest.
<b>DEL</b>	Löscht eine oder mehrere Dateien.
<b>DIR</b>	Listet die Dateien und Unterverzeichnisse eines Verzeichnisses auf.
<b>DISKCOMP</b>	Vergleicht den Inhalt von zwei Disketten.
<b>DISKCOPY</b>	Kopiert den Inhalt von einer Diskette auf eine andere Diskette.
<b>DOSKEY</b>	Bearbeitet Befehlseingaben, ruft Windows-Befehle zurück und erstellt Macros.
<b>ECHO</b>	Zeigt Meldungen an bzw. schaltet die Befehlsanzeige ein oder aus.
<b>ENDLOCAL</b>	Beendet den lokalen Gültigkeitsbereich von Umgebungsänderungen in einer Batchdatei.
<b>ERASE</b>	Löscht eine oder mehrere Dateien.
<b>EXIT</b>	Beendet das Programm CMD.EXE (Befehlsinterpreter).
<b>FC</b>	Vergleicht zwei oder mehr Sätze von Dateien und zeigt die Unterschiede an.
<b>FIND</b>	Sucht eine Zeichenkette in einer oder mehreren Datei(en).
<b>FINDSTR</b>	Sucht Zeichenketten in Dateien.
<b>FOR</b>	Führt einen angegebenen Befehl für jede Datei in einem Dateiensatz aus.
<b>FORMAT</b>	Formatiert einen Datenträger für die Verwendung mit Windows.
<b>FTYPE</b>	Zeigt die Dateitypen an, die bei den Zuordnungen für die entsprechenden Dateierweiterungen verwendet werden bzw. ändert sie.
<b>GETMAC</b>	Zeigt unter Windows XP SP2 die Mac-Adresse an.
<b>GOTO</b>	Setzt den Windows-Befehlsinterpreter auf eine markierte Zeile in einem Batchprogramm.
<b>GRAFTABL</b>	Ermöglicht Windows, Sonderzeichen im Grafikmodus anzuzeigen.
<b>HELP</b>	Zeigt Hilfeinformationen zu Windows-Befehlen an.
<b>IF</b>	Verarbeitet Ausdrücke in einer Batchdatei abhängig von Bedingungen.
<b>LABEL</b>	Erstellt, ändert oder löscht die Bezeichnung eines Volumes.
<b>MD</b>	Erstellt ein Verzeichnis
<b>MKDIR</b>	Erstellt ein Verzeichnis.
<b>MODE</b>	Konfiguriert ein Systemgerät.
<b>MORE</b>	Zeigt Ausgabe auf dem Bildschirm seitenweise an.
<b>MOVE</b>	Verschiebt ein oder mehrere Dateien von einem Verzeichnis in ein anderes.
<b>NET USE</b>	Verbindet einen Computer mit einer freigegebenen Ressource oder trennt die Verbindung und zeigt Informationen über die Verbindungen eines Computers an. Der Befehl steuert außerdem ständige Netzwerkverbindungen.
<b>PATH</b>	Legt den Suchpfad für ausführbare Dateien fest oder zeigt diesen an.
<b>PAUSE</b>	Hält die Ausführung einer Batchdatei an und zeigt eine Meldung an.

<b>POPD</b>	Wechselt zu dem Verzeichnis, das durch PUSHHD gespeichert wurde.
<b>PRINT</b>	Druckt eine Textdatei.
<b>PROMPT</b>	Ändert die Eingabeaufforderung.
<b>PUSHHD</b>	Speichert das aktuelle Verzeichnis, und wechselt dann zu einem anderen Verzeichnis.
<b>RD</b>	Entfernt ein Verzeichnis.
<b>RECOVER</b>	Stellt lesbare Daten von einem beschädigten Datenträger wieder her.
<b>REM</b>	Leitet Kommentare in einer Batchdatei bzw. CONFIG.SYS ein.
<b>REN</b>	Benennt eine Datei bzw. Dateien um.
<b>RENAME</b>	Benennt eine Datei bzw. Dateien um.
<b>REPLACE</b>	Ersetzt Dateien.
<b>RMDIR</b>	Löscht ein Verzeichnis.
<b>SET</b>	Setzt oder löscht die Umgebungsvariablen bzw. zeigt sie an.
<b>SETLOCAL</b>	Beginnt den lokalen Gültigkeitsbereich von Umgebungsänderungen in einer Batchdatei.
<b>SHIFT</b>	Verändert die Position ersetzbarer Parameter in Batchdateien.
<b>SORT</b>	Sortiert die Eingabe.
<b>START</b>	Startet ein eigenes Fenster, um ein bestimmtes Programm oder einen Befehl auszuführen.
<b>SUBST</b>	Weist einem Pfad einen Laufwerksbuchstaben zu.
<b>TASKLIST</b>	Zeigt alle zurzeit laufenden Aufgaben inklusive der Dienste an.
<b>TASKKILL</b>	Bricht einen laufenden Prozess oder eine Anwendung ab oder beendet ihn bzw. sie.
<b>TIME</b>	Zeigt die Systemzeit an bzw. legt sie fest.
<b>TITLE</b>	Legt den Fenstertitel für das Eingabeaufforderungsfenster fest.
<b>TREE</b>	Zeigt die Ordnerstruktur eines Laufwerks oder Pfads grafisch an.
<b>TYPE</b>	Zeigt den Inhalt einer Textdatei an.
<b>VER</b>	Zeigt die Windows-Version an.
<b>VERIFY</b>	Legt fest, ob überwacht werden soll, ob Dateien korrekt auf den Datenträger geschrieben werden.
<b>VOL</b>	Zeigt die Datenträgervolumenbezeichnung und die Seriennummer an.
<b>XCOPY</b>	Kopiert Dateien und Verzeichnisbäume.

## help in Kombination mit anderen Befehlen

In Kombination mit einem weiteren Batchbefehl z.B. `help md` erhält man weitere Informationen über diesen. Folgende Bildschirmausgabe erhält man bei der Eingabe des Befehls in der Kommandozeile unter WindowsXP:

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
```

```
C:\>help md
```

```
Erzeugt ein Verzeichnis.
```

```
MKDIR [Laufwerk:]Pfad
MD [Laufwerk:]Pfad
```

```
Wenn die Befehlsweiterungen aktiviert sind, wird MKDIR folgendermaßen
verändert:
```

```
MKDIR erzeugt, wenn nötig, jedes Zwischenverzeichnis. Wenn zum
```

Beispiel das Verzeichnis \a nicht existiert, dann entspricht

```
mkdir \a\b\c\d
```

der folgenden Befehlsfolge:

```
mkdir \a
chdir \a
mkdir b
chdir b
mkdir c
chdir c
mkdir d
```

Diese Folge von Befehlen muss angegeben werden, wenn die Befehlsweiterungen nicht aktiviert sind.

```
C:\>
```

## WebLinks

### Foren / Hilfe zur Batch-Programmierung finden

---

Am besten lassen sich Beispiele z.B. über Google mit einem oder mehreren Schlüsselwörtern finden: "batch" "cmd" "windows" "commandozeile" etc. Viele aktive Foren (z.B. Administrator.de) werden dann simultan von den Suchmaschinen durchsucht. Findet man dann immer noch keine Antwort, kann man die Frage in einem der gefundenen Foren selber stellen. Zur Zeit aktive Foren für Hilfestellungen bei der Erstellung von Batchdateien sind:

- Administrator.de

### Ungeordnete Links (Seiten mit tutorien, Referenzen, Beispielen etc.)

---

- Übersicht (der Unterschiede) "cmd Befehle" in Windows XP/Vista/7

[http://www.script-example.com/themen/cmd\\_Batch\\_Befehle.php](http://www.script-example.com/themen/cmd_Batch_Befehle.php)

- <http://www.heise.de/ct/03/16/136/default.shtml> c't 16/2003, S. 136: Windows-Befehlszeile
- <http://www.ss64.com/nt/index.html> NT/XP Batch Befehle (en)
- <https://web.archive.org/web/20071230045548/http://www.admini.de/batch-infos.htm> Umfangreiche Informationen über Batchprogrammierung unter NT/2000/XP
- <http://www.fpschultze.de/> Batch Scripting Site mit vielen Beispielen
- <http://sven-of-nine.de/site/doku.php/faq:batch> Umfangreiche Funktionssammlung für Batchprogrammierer
- <http://www.knowware.de/?cat=6.1&book=batch> Leicht verständliche Einführung
  - <http://download.knowware.de/batch.pdf> als PDF Download

- [http://publibz.boulder.ibm.com/cgi-bin/bookmgr\\_OS390/BOOKS/IEA2B510/...](http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/IEA2B510/...) ([http://publibz.boulder.ibm.com/cgi-bin/bookmgr\\_OS390/BOOKS/IEA2B510/CCONTENTS?SHELF=IEA2BK32&DN=SA22-7598-01&DT=20010626133938](http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/IEA2B510/CCONTENTS?SHELF=IEA2BK32&DN=SA22-7598-01&DT=20010626133938)) englischsprachiges JCL Benutzerhandbuch
  - <http://www.plogmann.net/w/2/54/index.htm> - Optionen des Befehls XCOPY
- <http://www.antonis.de/dos/#download/> Diverse Beispielskripte
- <http://www.easy-coding.de/shellscripting.html?s=5fb492fd4b515b0c880795a6a7d92ef5c4aed29c> Forum über Shellscripting
- <http://home.mnet-online.de/horst.muc/maind.htm> Eine Sammlung von Zusatztools für Batch-Skripter
- [http://www.axel-hahn.de/axel/page\\_compi/bat\\_index.htm](http://www.axel-hahn.de/axel/page_compi/bat_index.htm) Viele weitere Informationen und Tools zum Thema
- [http://www.script-example.com/themen/Windows\\_Batch.php](http://www.script-example.com/themen/Windows_Batch.php) Batch Dateien erstellen für absolute Beginner

## Tutorials

---

- [Batch for Runaways Part I \(https://www.administrator.de/Workshop\\_Batch\\_for\\_Runaways\\_-\\_Part\\_I\\_-\\_Beispiel\\_FindLongPath.Bat\\_Bedenklich\\_lange\\_Pfade\\_finden.html\)](https://www.administrator.de/Workshop_Batch_for_Runaways_-_Part_I_-_Beispiel_FindLongPath.Bat_Bedenklich_lange_Pfade_finden.html) - Tutorial Batch I
- [Batch for Runaways Part II \(https://www.administrator.de/Workshop\\_Batch\\_for\\_Runaways\\_-\\_Part\\_II\\_-\\_Ein\\_bisschen\\_Handwerkszeug.html\)](https://www.administrator.de/Workshop_Batch_for_Runaways_-_Part_II_-_Ein_bisschen_Handwerkszeug.html) - Tutorial Batch II
- [Batch for Runaways Part III \(https://www.administrator.de/Workshop\\_Batch\\_for\\_Runaways\\_-\\_Part\\_III\\_-\\_Datums-\\_und\\_Zeitvariablen\\_im\\_Batch.html\)](https://www.administrator.de/Workshop_Batch_for_Runaways_-_Part_III_-_Datums-_und_Zeitvariablen_im_Batch.html) - Tutorial Batch III
- Tutorials auf Administrator.de
  - [Die For-Schleife \(https://www.administrator.de/Tutorial\\_zur\\_FOR-Schleife.html\)](https://www.administrator.de/Tutorial_zur_FOR-Schleife.html) - Tutorial "FOR"
  - [Variablen \(https://www.administrator.de/Wie\\_Arbeite\\_ich\\_mit\\_Batch\\_UmgebungsVariable\\_n%3F\\_Erstellung\\_Umgang\\_Erweiterungen\\_Ver%C3%A4nderungen.html\)](https://www.administrator.de/Wie_Arbeite_ich_mit_Batch_UmgebungsVariable_n%3F_Erstellung_Umgang_Erweiterungen_Ver%C3%A4nderungen.html) - Tutorial Umgebungsvariablen

## Bücher

---

- [Windows Command-Line Administrator's Pocket Consultant, 2nd Edition \(Amazon-Link\) \(https://www.amazon.com/Windows-Command-Line-Administrators-Pocket-Consultant/dp/0735622620\)](https://www.amazon.com/Windows-Command-Line-Administrators-Pocket-Consultant/dp/0735622620)
- Wikibooks (eng.): [Windows Batch Scripting \(https://en.wikibooks.org/wiki/Windows\\_Batch\\_Scripting\)](https://en.wikibooks.org/wiki/Windows_Batch_Scripting)

## Referenzen

---

Referenzen erlauben einen Überblick über vorhandene Befehle. Primär für erfahrene Anwender sind sie im Regelfall sehr kompakt enthalten, aber oftmals mit ausführlichen Erklärungen als Verlinkung mit einem einzelnen Befehl.

- [Microsoft Technet \(http://technet.microsoft.com/en-us/library/cc722159.aspx\)](http://technet.microsoft.com/en-us/library/cc722159.aspx) - Windows Vista - Command-Line Tools Technical Reference (en)
- [Microsoft Technet \(http://technet.microsoft.com/en-us/library/bb490890.aspx\)](http://technet.microsoft.com/en-us/library/bb490890.aspx) - Windows XP - Command-line reference A-Z (en)

- [Microsoft Product Documentation \(http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/ntcmds.msp?mfr=true\)](http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/ntcmds.msp?mfr=true) - Windows XP Professional Product Documentation
- [Microsoft Technet \(http://technet.microsoft.com/de-de/library/cc785423.aspx\)](http://technet.microsoft.com/de-de/library/cc785423.aspx) - Windows Server 2003 - Befehlszeilenreferenz (de)
- [Microsoft Technet \(http://technet.microsoft.com/en-us/library/cc754340.aspx\)](http://technet.microsoft.com/en-us/library/cc754340.aspx) - Windows Server 2008 - Command Reference (en)

---

Abgerufen von „[https://de.wikibooks.org/w/index.php?title=Batch-Programmierung/\\_Druckversion&oldid=962178](https://de.wikibooks.org/w/index.php?title=Batch-Programmierung/_Druckversion&oldid=962178)“

---

**Diese Seite wurde zuletzt am 2. Juni 2021 um 08:18 Uhr bearbeitet.**

Der Text ist unter der Lizenz Creative Commons Namensnennung – Weitergabe unter gleichen Bedingungen verfügbar. Zusätzliche Bedingungen können gelten. Einzelheiten sind in den Nutzungsbedingungen beschrieben.