Kurze Einführung in MS-DOS

Diese Einführung ist eigentlich uralt - aus den Zeiten, als die Comuter noch mittels kryptischer Kommando bedient wurden und keine grafische Bedienoberfläche hatten. Erstaunlicherweise hat MS-DOS alle bisherigen Windows-Versionen überlebt, angefangen von Windows 3.0 bis heute - und zwar in Form der sogenannten Eingabeaufforderung. Startet man diese, erscheint ein Terminalfenste mit schwarzem Hintergrund, das sich wie der Bildschirm des guten alten DOS verhält.

Erstaunlicherweise können sich heute wieder viele für die Möglichkeit begeistern, den PC per Kommandozeile zu steuern und dort Dinge zu erledigen, die unter Windows nur umständlich zu realisieren sind. Beispielsweise ein Backup von ganzen Ordner per `xcopy`-Kommando oder der Aufruf von `ping` oder `tracert`. Auch eine sehr primitive Kommandosprache ist vorhanden: es lassen sich Befehle in "Batch-Dateien" zusammenfassen und durch Starten der Batchdatei (Endung ".bat") en bloc ausführen.

Deshalb habe ich diese alte Anleitung wieder herausgekramt. Sie soll einen Kurzüberblick über MS-DOS geben. Mit dem Befehl `help <Kommandoname>` erfahren Sie mehr über das entsprechende Kommando.

Ein Kommando ist der Aufruf eines Programms (externes Kommando) oder eines internen DOS-Befehls (enthalten im Kommandointerpreter COMMAND.COM). Einige dieser Kommandos werden weiter hinten besprochen. Bei der Eingabe von Kommandos spielt die Gross/Kleinschreibung keine Rolle. Die Syntax ist bei allen Kommandos gleich:

```
<Kommandoname> <Parameter> <Parameter> ...
```

Die Parameter können entweder Dateinamen oder "Schalter" sein. Sie werden durch Leerzeichen voneinander getrennt. Zum Beispiel:

```
copy A:DATEI B:DATEI /V
        |       |    |
        Dateinamen Schalter
```

Bei vielen Kommandos können mehrere Dateien gleichzeitig bearbeitet werden. Dazu können sogenannte Jokerzeichen (Wildcards) im Dateinamen aufgeführt werden:

- Der * steht für für beliebige weitere Zeichen im Dateinamen
- Das ? steht für ein beliebiges Zeichen im Dateinamen

DATEI? deckt sich also z. B. mit DATEI1, DATEI2, DATEI3 - nicht aber mit DATEI12. DAT* deckt sich mit allen Namen die mit DATEI beginnen, also z. B. mit DATEI, DATEI1, DATEN, DATA, etc. *.* deckt sich mit allen Dateinamen.

Gerätenamen

Damit einzelne Kommandos anstelle von Dateien auch Geräte (z. B. den Drucker) ansprechen können, haben die Geräte festgelegte Namen, die jederzeit anstelle von Dateinamen eingesetzt werden können. Mit copy A:DATEI PRN: kann man eine Datei z. B. auf dem Drucker ausgeben. Gültige Gerätenamen sind:

```
CON: Ein-/Ausgabe über Bildschirm und Tastatur
PRN: Standard-Druckerausgabe (1. Druckerschnittstelle)
LPT1: 1. Druckerschnittstelle
LPT2: 2. Druckerschnittstelle
LPT3: 3. Druckerschnittstelle
COM1: 1. serielle Schnittstelle
COM2: 2. serielle Schnittstelle
NUL: leeres Pseudogerät ("schwarzes Loch" für Ausgaben)
```

Laufwerke

Disketten- und Festplattenlaufwerke werden durch einen einzigen Buchstaben identifiziert. Dem Buchstaben folgt zur Unterscheidung von Dateinamen ein Doppelpunkt. Ein physikalisches Laufwerk kann mehrere Laufwerksnamen haben. Wenn Sie z. B. noch ein Diskettenlaufwerk im Computer haben, kann dieses Laufwerk unter A: und B: angesprochen werden.

Der Wechsel des Laufwerks erfolgt auf Kommandoebene durch Eingabe des Laufwerksnamens. Das aktuell eingestellte Laufwerk wird bei der Bereitschaftsanzeige am Bildschirm ausgegeben:

`C:\>`	aktuell ist Laufwerk C:
`F:`	Kommando zum Laufwerkswechsel
`F:\>`	neue Bereitschaftsanzeige

Dateiverzeichnisse

Auf einer Festplatte mit 100 MByte Kapazität oder mehr haben viele Dateien Platz. Um die Übersicht zu behalten muss man seine Dateien strukturieren. Dateiverzeichnisse (Directories, Unterverzeichnisse, Kataloge) bieten die Möglichkeit der Strukturierung der Dateien. In einem Verzeichnis werden logisch zusammenhängende Dateien und Programme abgelegt. Von "aussen" ist nur der Katalogname zu sehen. Verzeichnisse lassen sich "schachteln", d. h. ein Verzeichnis kann seinerseits Unterverzeichnisse enthalten.

- Eine Datei in einem Verzeichnis kann über den **Pfad** angesprochen werden. So lässt sich die Datei BRIEF.TXT im Verzeichnis BRIEFE des Hauptverzeichnisses LEHMANN über die Pfadangabe `LEHMANN\BRIEFE\BRIEF.TXT` ansprechen.

- Die einzelnen Verzeichnisnamen werden durch den \ getrennt. Dabei wird in der Regel von dem Verzeichnis ausgegangen, in dem man sich befindet (relativer Pfad). Durch Voranstellen des \ vor den Pfad wird vom Hauptverzeichnis (Root) ausgegangen (absoluter Pfad).

- Wenn man sich z. B. im Verzeichnis LEHMANN befindet, kann man die o. g. Datei BRIEF.TXT über den relativen Pfad BRIEFE\BRIEF.TXT erreichen. Der absolute Pfad für BRIEF lautet dagegen \LEHMANN\BRIEFE\BRIEF.

- In jedem Verzeichnis gibt es immer zwei spezielle Unterverzeichnisse mit den Namen "." und "..".
  `.` ist ein Verweis auf das aktuelle Verzeichnis.
  `..` ist ein Verweis auf das übergeordnete Verzeichnis.

  Mit dem Kommando `cd ..` (siehe unten) gelangt man also eine Stufe zurück in Richtung Hauptverzeichnis. Befindet man sich z. B. im Verzeichnis \LEHMANN\BRIEFE, erreicht man mit `cd ..` das Verzeichnis \LEHMANN.

Kommandos zur Arbeit mit Unterverzeichnissen:

- `dir` Anzeige aller Dateien im aktuellen Verzeichnis. `dir` hat zwei mögliche Parameter. Der erste Parameter ist ein Dateiname (`dir *.*` liefert alle Dateien (`dir .` übrigens auch); `dir *.txt` liefert nur die Text-Dateien). Der zweite Parameter steuert die Ausgabe: mit /W erhalten Sie eine kompakte Liste; /S listet auch alle Unterverzeichnisse und darin befindliche Dateien auf. Weitere Parameterbuchstaben werden durch `dir /?` angezeigt.

- `mkdir, md`: Erzeugen eines neuen Unterverzeichnisses. Das Kommando hat den neuen Verzeichnisnamen als Parameter; z. B. `mkdir foobar`

- `rmdir, rd`: Löschen eines Unterverzeichnisses. Das Verzeichnis wird nur gelöscht, wenn es keine Dateien mehr enthält.

- `chdir, cd`: Wechseln in ein beliebiges Unterverzeichnis. Als Parameter wird der Pfad zu diesem Verzeichnis angegeben; z. B. `cd meyer\daten`

Der Prompt-Befehl

Dieser Befehl verändert die Bereitschaftsanzeige (Prompt). Da er keinen Schaden anrichten kann, lässt sich wunderbar damit spielen. Durch den Befehl `prompt` ohne Parameter wird der Urzustand wiederhergestellt. Parameter ist eine beliebige Zeichenkette, die dann als Bereitschaftsanzeige verwendet wird. So führt der Befehl `prompt Nächstes Kommando:` zur Bereitschaftsanzeige:

```
Nächstes Kommando:
```

Deshalb wird die Bereitschaftssanzeige auch kurz als "Prompt" bezeichnet und vielfach läuft die gesamte Eingabeaufforderung unter der Bezeichnung "DOS-Prompt". Es gibt eine Reihe von Spezialzeichenkombinationene für Sonderfunktionen, die alle mit einem "$"_Zeichen beginnen:

Zeichen	Anzeige
$$	$
$_	Zeilenvorschub
$b	
$d	Anzeige des Datums
$e	Das ESC-Zeichen
$g	>
$h	Rückschritt ein Zeichen (zum Löschen von Zeichen)
$l	<
$n	Das aktuelle Laufwerk
$p	Das aktuelle Verzeichnis
$q	=
$s	Leerzeichen
$t	Die Uhrzeit
$v	Die DOS-Version

Als Standardprompt ist meist `prompt $p$g` eingestellt. Beispielsweise würde `prompt $n$p$s$g` immer das aktuelle Laufwerk und das aktuelle Verzeichnis anzeigen. `prompt $d$s$s$t$_$n$p$s$g` zeigt in der ersten Zeile Datum und Uhrzeit und in der folgenden Zeile Laufwerk und Verzeichnis an.

Kommandos zur Arbeit mit Disketten und Dateien und Info-Kommandos

`attrib`	Schreibschutz setzen/löschen und andere Attribute ändern
`chkdsk`	Platte überprüfen (veraltet)
`copy`	Datei(en) kopieren
`comp`	Vergleicht den Inhalt zweier Dateien oder Sätze von Dateien
`del`	Datei(en) löschen
`dir`	Anzeige von Dateien (siehe oben)
`find`	Suchen in Dateien
`format`	Formatiert einen Dateinträger für die Verwendung mit Windows
`mklink`	Erstellt symbolische Links und feste Links

more	Zeigt Ausgabe auf dem Bildschirm seitenweise an
move	Verschiebt ein oder mehrere Dateien von einem Verzeichnis in ein anderes
print	Druckt eine Textdatei oder gibt sie aus
rename, ren	Dateien umbenennen
sort	Sortiert die Eingabe
subst	Ordnet einen Pfad einem Laufwerkbuchstaben zu
systeminfo	Zeigt computerspezifische Eigenschaften und Konfigurationen an
tasklist	Zeigt alle zurzeit laufenden Aufgaben inklusive der Dienste an
tree	Verzeichnisstruktur anzeigen (Erweiterung von dir)
type	Datei ausgeben
ver	Zeigt die Windows-Version an
vol	Zeigt die Volumebezeichnung und die Seriennummer des Datenträgers an
xcopy	Erweitertes Kopierprogramm

Bei der Besprechung der wichtigsten Kommandos auf den folgenden Seiten werden nur die am häufigsten verwendeten Parameter erwähnt. Für eine vollständige Beschreibung der Kommandos sehen Sie bitte im Handbuch nach. Alternativ können Sie das Kommando mit dem Parameter "/?" aufrufen oder das Kommando `help <Kommandoname>` verwenden.

Das copy-Kommando

ist das wohl am häufigsten gebrauchte Kommando. Es dient zum Übertragen von Dateien zwischen verschiedenen Laufwerken und Verzeichnissen. Mit `copy` ist auch die Ausgabe von Dateien auf Geräten und das Zusammenführen von Dateien möglich.

copy arbeitet nur innerhalb eines Verzeichnisses. Will man ganze Verzeichnisse einschließlich der Unterverzeichnisse kopieren, muss man `xcopy` verwenden. Aufruf: `copy Quellpfad Zielpfad`. Besteht die Angabe des Zielpfades nur aus dem Datenamen, ist das aktuelle Laufwerk/Verzeichnis Zielverzeichnis. Fehlt die Zielangabe ganz, erhalten die neuen Dateien den gleichen Namen wie die Quelldateien.

Beispiele (mit Erläuterung):

`copy C:DATEN D:DATEN` Kopiert DATEN von Laufwerk C: nach D:
`copy C:DATEN D:` Dasselbe, nur kürzer geschrieben
`copy DATEN DATEN.BAK` Anlegen einer Backup-Datei
`copy C:TEXT` Kopieren der Datei TEXT auf Laufwerk C in das aktuelle Laufwerk
`copy C:*.BAT D:` Kopieren aller Batch-Dateien nach Laufwerk D:
`copy C:*.*` Kopieren aller Dateien von Laufwerk C: auf das aktuelle Laufwerk
`copy \TEXTE\BRIEF \INFOS` Kopieren der Datei BRIEF aus dem Verzeichnis TEXTE in die Datei INFOS im Hauptverzeichnis. Wenn INFOS ein Verzeichnis ist, wird in INFOS eine Datei BRIEF angelegt (zweideutig!!!).
`copy T1+T2+T3 T.GES` Die Dateien T1, T2 und T3 werden in der Datei T.GES vereinigt.
`copy *.TX1+*.TX2 *.TXT` Es werden jeweils die Dateien gleichen "Vornamens" zusammenkopiert (T1.TX1+T1.TX2 → T1.TXT sowie T2.TX1+T2.TX2 → T2.TXT, usw.
`copy CON: TEST` Die Tastatureingabe wird in die Datei TEST kopiert. Die Eingabe wird mit Strg-Z beendet.

Das xcopy-Kommando

Kopiert Dateien und komplette Verzeichnisstrukturen. Das Kommando hat fast 30 Parameter, weshalb hier nur die wichtigsten aufgeführt werden. Der generelle Aufruf lautet `xcopy Quellpfad Zielpfad Parameter`:

```
/A        kopiert nur Dateien mit gesetztem Archivattribut, ändert das Attribut nicht.
/M        kopiert nur Dateien mit gesetztem Archivattribut und löscht das Attribut.
/D:M-T-J  kopiert nur die an oder nach dem Datum geänderten Dateien.
          ist kein Datum angegeben, werden nur Dateien kopiert,
          die neuer als die bestehenden Zieldateien sind.
/S        kopiert Verzeichnisse und Unterverzeichnisse, die nicht leer sind.
/E        kopiert alle Unterverzeichnisse (leer oder nicht leer).
/Q        zeigt beim Kopieren keine Dateinamen an (quiet).
/F        zeigt die Namen der Quell- und Zieldateien beim Kopieren an.
/H        kopiert auch Dateien mit den Attributen 'Versteckt' und 'System'.
/R        überschreibt schreibgeschützte Dateien.
/U        kopiert nur Dateien, die im Zielverzeichnis vorhanden sind.
/K        kopiert alle Attribute. Standardmäßig wird 'Schreibgeschützt' gelöscht.
```

Dank seiner vielen Parameter kann `xcopy` für die Datensicherungen benutzt werden. Seit Windows Vista muss man die Eingabeaufforderung "als Administrator ausführen", um diesen Befehl nutzen zu können. Bei sehr umfangreichen Kopieraufträgen ist `xcopy` unzuverlässig. Das Programm erstellt zu Arbeitsbeginn eine Liste aller zu kopierenden Dateien. Wenn die Anzahl der Dateien zu groß ist, passt diese Liste nicht in den Speicher. Der Kopiervorgang ist dann unvollständig.

```
xcopy [QUELLE] [ZIEL] /S /E /C /H /O /R /Y /D /V
```

Mit diesem Befehl werden alle Unterverzeichnise (/S), unabhängig davon, ob sie leer sind oder nicht (/E), von [QUELLE] zu [ZIEL] kopiert. Hierbei wird das Kopieren auch dann fortgesetzt, wenn Fehler auftauchen (/C) oder die Dateien eigentlich als "versteckt" oder "System" gekennzeichnet sind (/H). Zusammen mit der Datei werden auch alle Schreib- und Benutzerrechte mit kopiert, die für diese Datei im System hinterlegt sind (/O).
Ist im Zielverzeichnis bereits eine alte Version des Backups vorhanden, soll `xcopy` sie auch dann überschreiben, wenn sie schreibgeschützt sind (/R). Zur Vermeidung von ständigen Fragen wird (/Y) angegeben. Es sollen dabei nur die jeweils aktuelleren Dateien erhalten bleiben (/D). Als zusätzliche Absicherung kann `xcopy` auch alle kopierten Dateien überprüfen (/V).

Für umfangreiche Kopierarbeiten wurde ein Nachfolger entwickelt. Das Programm `robocopy` ist deutlich robuster als `xcopy`. In Windows Vista und Nachfolgern gehört es zum Betriebssystem, sogar mit Erläuterungen in deutscher Sprache.

Das Attribut-Kommando

Das Kommando `attrib` zeigt Dateiattribute an oder ändert sie. Es werden die Attribute "schreibgeschützt", "archiv", "system" und "versteckt", die Dateien oder Verzeichnissen zugewiesen wurden, angezeigt. Di Syntax lautet:

```
ATTRIB [+R | -R] [+A | -A ] [+S | -S] [+H | -H] [+I | -I] [Pfad] [/S [/D] [/L]]
```

Wird nur ein Pfad angegeben, zeigt das Kommando alle Dateien und deren Attribute an. Bei Angae von Attributwerten setzt "+" ein Attribut und "-" löscht ein Attribut. Es gibt die folgenden Attribute:

```
R   Attribut für 'schreibgeschützte Datei'
A   Attribut für 'zu archivierende Datei'
S   Attribut für 'systemdatei'
H   Attribut für 'versteckte Datei'
I   Attribut für 'inhalt nicht indiziert'
```

Hinter der Pfadangabe (Verzeichnis oder Datei) können noch folgende Parameter angegeben werden:

```
/S        verarbeitet passende Dateien im aktuellen Verzeichnis und in allen Unterverzeichnissen.
/D        verarbeitet auch die Verzeichnisse selbst.
/L        verarbeitet die Attribute des symbolischen Links anstelle des Linkziels.
```

Das find-Kommando

Sucht in einer oder mehreren Dateien nach einer Zeichenfolge. Die Syntax lautet:

```
FIND [/V] [/C] [/N] [/I] "Zeichenfolge" [Pfad]
```

Zu beachten ist, dass die Parameter vor dem Suchbegriff eingefügt werden müssen. Es gibt vier Parameter:

```
  /V        Zeigt alle Zeilen an, die die Zeichenfolge NICHT enthalten.
  /C        Zeigt nur die Anzahl der die Zeichenfolge enthaltenden Zeilen an.
  /N        Zeigt die Zeilen mit ihren Zeilennummern an.
  /I        Ignoriert Groß-/Kleinschreibung bei der Suche.
```

Fehlt der Pfad, so durchsucht FIND von der Tastatur aus eingegebenen Text oder die Ausgabe des Befehls, der FIND in einer Pipe (Befehlsverkettung) vorangestellt ist.

Eine wesentlich erweiterte Suche erlaubt das Programm `findstr`, das mit regulären Ausdrücken arbeitet (ähnlich dem `grep` bei Unix/Linux).

Ein- und Ausgabeumleitung

Bildschirm-Ausgaben und Tastatur-Eingaben lassen sich auf Dateien oder Geräte umleiten. Das funktioniert auf der Kommandozeile und innerhalb von Batch-Dateien (siehe unten). Die Umleitung erfolgt über folgende Symbole:

- **>** Ausgabe des Programms in Datei oder an ein Gerät
- **<** Eingabe des Programms aus Datei oder von einem Gerät
- **|** Ausgabe eines Programms direkt in die Eingabe eines anderen leiten

Umgeleitet werden können jedoch nur die Standardausgabe und die Standardeingabe von DOS. Die Standard-Fehlerausgabe wird weiterhin im DOS-Fenster ausgegeben. Ob und welche Ausgaben eines Programmes sich umleiten lassen, muss man gegebenenfalls durch Ausprobieren herausfinden.

Die Ausgabeumleitung wird häufig dann eingesetzt, wenn man den Output eines Programms anderweitig weiterverarbeiten will, z. B. in einer Dokumentation. Bei der Ausgabeumleitung gibt es noch eine weitere Besonderheit: Ein einfaches ">" verwendet immer eine leere Datei. Falls die Datei schon existierte, wird sie überschreiben. Man kann jedoch mittels ">>" Informationen am Ende einer existierenden Datei anhängen. Dazu einige Beispiele:

```
echo "Hallo Welt" > test                Schreibt die Zeile "Hallo Welt"
                                        in die Datei "test"
dir >> test                             Hängt die Verzeichnisliste an
                                        die Datei "test" an
ech "fertig" >> test                    Schreibt "Fertig" als letzte
                                        Zeile in die Datei "test"
```

Geräte-Namen koennen unter DOS (fast) wie Dateinamen verwendet werden. Die Ausgabe kann also auch z. B. an die serielle Schnittstelle geschickt werden:

```
copy daten.txt > :COM1
```

NUL ist ein Pseudo-Gerät, das aber benutzt werden kann, um Ausgaben ins Nichts zu schicken. Damit lassen sich z. B. unerwünschte Ausgaben in Batch-Dateien unterdrücken, sofern die Standard-Ausgabe verwendet wird.

```
copy datei.txt D: >NUL              unterdrückt die Meldung
                                       " nn Datei(en) kopiert"
```

Wenn das entsprechende Programm nichts ausgibt, wird trotzdem die angegebene Datei erstellt. Eine Umleitung beim Aufruf einer Batch-Datei ist wirkungslos. Wer auf diese Weise ALLE Ausgaben einer Batch-Datei umleiten will, muss dazu eine weitere Inkarnation des Kommandointerpreters starten, z. B. durch `cmd.exe /C tuwas.bat > datei`.

Wie bei der Ausgabe-Umleitung ist auch bei der Eingabe eine Umleitung nur möglich, wenn das Programm von der Standardeingabe liest. Normalerweise handelt es sich hierbei um die Tastatur. Quelle für eine

Eingabeumleitung kann auch ein Gerät sein, z. B. COM1.

Die Eingabe-Umleitung ist das Gegenstück zur Ausgabe-Umleitung. Dabei werden Eingaben, die eigentlich von der Tastatur erwartet werden, aus einer Datei eingelesen. Zum Beispiel:

```
more < ein_langer.txt
```

Das DOS-Programm `more` gibt den Inhalt der Datei auf dem Bildschirm aus, und zwar mit Stopp nach jeder vollen Seite.

Als letzte Möglichkeit kann die (Standard-)Ausgabe eines Programms mittels "|" in die (Standard-)Eingabe eines anderen gleitet werden, zum Beispiel:

```
foobar | sort                Ausgabe des Programms "foobar"
                             sortiert ausgeben
echo J | del C:\*.*          Nachfrage "wollen Sie wirklich..."
                             automatisch beantworten
```

Das "|" wird auch "Pipe" (Röhre) genannt und bewirkt, dass die Ausgabe von der linken Seite als Eingabe für die rechte Seite verwendet wird. DOS produziert dazu zwar insgeheim eine Zwischendatei (im Verzeichnis %TEMP%), aber man braucht sich um deren Namen und um das anschließende Löschen nicht zu kümmern.

Es lassen sich mehrere Umleitungen hintereinander schalten. Für die richtige Reihenfolge gilt, dass mit "|" (Pipe) verbundene Programme immer von links nach rechts abgearbeitet werden. Befehlsparameter und Optionen müssen unmittelbar auf den zugehörigen Befehl folgen.

Programme, die von der Standardeingabe lesen und auf die Standardausgabe schreiben, nennt man auch "Filter". Es gibt drei Filter unter den DOS-Kommandos:

- `more` hält die Bildschirmausgabe alle 24 Zeilen an und ermöglicht so das Blättern
- `find` (siehe auch oben) erlaubt das Suchen in Textdateien. Das Programm kann auch wie ein "normales" Kommando verwendet werden.
- `sort` ist ein Sortierprogramm für Texte. Zwei Parameter steuern die Sortierung: `/+nn` legt die Spalte fest, ab der sortiert wird (z. B. `/+12`); `/R` sortiert absteigend. Leider beachtet SORT weder die Gross/Kleinschreibung noch die deutschen Umlaute.

Batch-Betrieb

Der Batch-Betrieb (Stapelverarbeitung) erlaubt die Zusammenfassung mehrerer Kommandos, die hintereinander ausgeführt werden sollen in einer Batch-Datei (Befehls-Datei). Die Befehlsdatei wird durch Eingabe ihres Namens - also auf die gleiche Weise wie ein einzelnes Kommando - aufgerufen.

Über Parameter, die beim Aufruf angegeben werden können, lässt sich der Ablauf der Batch-Datei steuern. Zusätzliche Stapelbefehle erlauben eine komfortablere und flexiblere Gestaltung des Ablaufs. Dadurch, dass sich Programmparameter in der Stapeldatei fest einstellen lassen, macht das Batch-System die Bedienung des Computers für den Benutzer einfacher. Eine Batch-Datei hat immer die Endung ".bat" und wird als Textdatei mit einem Editor erstellt.

Ein erstes Beispiel:

Hat man seine Festplatte ordentlich in Unterverzeichnisse unterteilt, ist zum Aufruf einer Applikation, z. B. der Fakturierung eine Folge von Kommandos nötig. Diese Kommandofolge schreibt man in eine Batchdatei und kann so mit einem Befehl (FAKT) die Applikation aufrufen:

```
cd FAKTURA                   Wechsel ins Verzeichnis
fbackup.exe                  Backup machen
faktura.exe                  Aufruf des Programms
cd \                         zurück ins Hauptverzeichnis
```

Neben dem rein sequentiellen Ablauf können auch Bedingungen und Schleifen programmiert werden. Für diesen Zweck besitzt das DOS spezielle Batch-Kommandos (die Liste ist gekürzt, eine komplette Kommandoliste liefert der Befehl HELP):

`BREAK`	Schaltet die erweiterte Überprüfung für STRG+C ein bzw. aus
`CALL`	Ruft eine Batchdatei von einer anderen Batchdatei aus auf
`CLS`	Löscht den Bildschirminhalt
`CHOICE`	Erlaubt die Eingabe von Optionen, die dann den Errorlevel setzen
`COLOR`	Legt die Hintergrund- und Vordergrundfarben für die Konsole fest
`DATE`	Zeigt das Datum an bzw. legt es fest
`ECHO`	Zeigt Meldungen an bzw. schaltet die Befehlsanzeige ein oder aus
`EXIT`	Beendet die Eingabeaufforderung (Befehlsinterpreter CMD.EXE)
`FOR`	Führt einen angegebenen Befehl für jede Datei in einem Dateiensatz aus
`GOTO`	Springt zu einer markierte Zeile in einem Batchprogramm
`IF`	Verarbeitet Ausdrücke in einer Batchdatei abhängig von Bedingungen
`PAUSE`	Hält die Ausführung einer Batchdatei an und zeigt eine Meldung an
`REM`	Leitet Kommentare in einer Batchdatei ein
`SET`	Setzt oder löscht die Umgebungsvariablen bzw. zeigt sie an
`SHIFT`	Verändert die Position ersetzbarer Parameter in Batchdateien
`SHUTDOWN`	Ermöglicht Herunterfahren des Computers
`START`	Startet ein bestimmtes Programm in einem neuen Fenster
`TASKLIST`	Zeigt alle zurzeit laufenden Aufgaben inklusive der Dienste
`TASKKILL`	Beendet einen laufenden Prozess oder eine Anwendung
`TIME`	Zeigt die Systemzeit an bzw. legt sie fest

Übergabeparameter und `echo`

Wie bei einem Programm kann man auch einem Batch-Job Parameter übergeben. Diese Parameter werden in der Batch-Datei durch die Platzhalter `%0` bis `%9` dargestellt. Der Platzhalter `%0` enthält den Namen der Batch-Datei. Die folgende Beispieldatei enthält die Zeilen:

```
echo Ich bin die Batchdatei %0
echo Inhalt von %1
dir /W %1
```

Der Aufruf `t1.bat D:\test` führt im Batch-Job zu folgender Ausgabe:

```
D:\>echo Ich bin die Batchdatei t1.bat
Ich bin die Batchdatei t1.bat

D:\>echo Inhalt von D:\test
Inhalt von D:\test

D:\>dir /W D:\test
 Datenträger in Laufwerk D: ist DATA
 Volumeseriennummer: 225B-A85F

 Verzeichnis von D:\test

[.]          [..]         bild1.jpg    bild2.jpg    bild3.jpg    bild4.jpg
bild5.jpg    bild6.jpg    bild7.jpg    dd0.pdf      index.html   [t1]
```

```
[t2]         [t3]         [t4]         [t5]
               9 Datei(en),      1.522.271 Bytes
```

So ganz befriedigend ist der Output nicht, weil alle ausgeführten Kommandos geechot werden. Das mag beim Test ganz praktiksch sein, bei der endgültigen Batchdatei will man das aber meist nicht.

`echo` gibt normalerweise aslles aus, was hinter dem Kommando in der Eingabezeile steht. Durch `echo off` kann hingegen das Echo der Kommandos abgeschaltet und durch `echo off` wieder eingeschaltet werden. Ist die Anzeige abgeschaltet, kann man nach dem Wort `echo` einen beliebigen Text plazieren, der dann auf dem Bildschirm ausgegeben wird. Üblicherweise wird das Echo in Batch-Dateien gleich zu Anfang abgeschaltet. Die geänderte Batchdatei sieht dann folgendermaßen aus:

```
echo off
echo Ich bin die Batchdatei %0
echo Inhalt von %1
dir /W %1
```

Der Aufruf `t1.bat D:\test` führt im Batch-Job zu folgender verbesserten Ausgabe:

```
F:\>echo off
Ich bin die Batchdatei t1.bat
Inhalt von D:\test
 Datenträger in Laufwerk D: ist DATA
 Volumeseriennummer: 225B-A85F

 Verzeichnis von D:\test

[.]          [..]         bild1.jpg    bild2.jpg    bild3.jpg    bild4.jpg
bild5.jpg    bild6.jpg    bild7.jpg    dd0.pdf      index.html   [t1]
[t2]         [t3]         [t4]         [t5]
               9 Datei(en),      1.522.271 Bytes
               7 Verzeichnis(se), 73.434.910.720 Bytes frei
```

Jetzt stört eigentlich nur das Echo des `echo off`. Damit aber dieses Kommando nicht selbst trotzdem dargestellt wird, wird ihm ein Klammeraffe (@) vorgestellt, also `@echo off`. Das Voranstellen des Klammeraffen, um das Echo abzuschalten, funktioniert natürlich bei jedem Kommando. Die Beispieldatei lautet dann endgültig:

```
@echo off
echo Ich bin die Batchdatei %0
echo Inhalt von %1
dir /W %1
```

Ohne jeden Parameter aufgerufen gibt `echo` den jeweiligen Zustand (ON oder OFF) aus, nicht etwa eine Leerzeile. Um eine Leerzeile auszugeben müssen Sie das `echo`Kommando mit direkt folgendem Punkt angeben, also "`echo.`".

Eigentlich waren wir ja noch bei den Platzhaltern, die aus einem Prozentzeichen mit der laufenden Nummer des Befehlsparameters bestehen, also %1 ... %9. Es ist nur eine Ziffer möglich.

Was aber, wenn es mehr als neun Parameter gibt? Dann hilft der `shift`-Befehl. `shift` verschiebt alle Parameter nach links, d. h. %0 fällt heraus, %1 wird zu %0, %2 wird zu %1 und so weiter. `shift` kann bei Bedarf wiederholt werden. So kann auch eine variable Zahl von Parametern verarbeitet werden.

Umgebungsvariable und `set`

In Batch-Dateien lassen sich Variable benutzen, die als sogenannte Umgebungsvariable (auch "Shellvariable") in einem besonderen Speicherbereich des Kommandointerpreters liegen. Eine Variable wird durch eine Wertzuweisung definiert. Jede Werzuweisung erfolgt in der Form

```
set <Variablenname>=<Zeichenfolge (String)>   , z. B.:
set PROMPT=$p$g
set TEMP=D:\TMP
```

Der aktuelle Inhalt dieses Speichers kann jederzeit mit dem Kommando `set` (ohne Parameter!) auf der DOS-Befehlsebene aufgelistet werden.

Der Kommandointerpreter bedient sich der Umgebungsvariablen, um bestimmte Informationen zu ermitteln, beispielsweise die Verzeichnispfade, in denen Programme gesucht werden sollen (PATH) oder die Darstellung des Prompt (PROMPT). Alle Programme können diese Informationen nutzen. Durch Angabe des Variablennamens, eingeschlossenen in Prozentzeichen, kann der aktuelle Wert an jeder beliebigen Stelle der Batch-Datei verwendet werden:

```
D:\>echo %PATH%
C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;...

F:\>echo %PROMPT%
$P$G

F:\>echo %windir%
C:\Windows
```

Wenn die Befehlserweiterungen aktiviert sind (ab Windows 7 obligatorisch), gibt es einige Umgebungsvariablen, die aber nicht in der mit SET angezeigten Liste von Variablen auftauchen. Diese Variablenwerte dynamisch berechnet. Falls eine eigene Variable mit einem dieser Namen definiert wird, dann überschreibt diese Definition die unten stehende dynamische Definition:

`%CD%`	expandiert zum aktuellen Verzeichnisnamen.
`%DATE%`	expandiert zum aktuellen Datum (Format wie DATE-Befehl).
`%TIME%`	expandiert zur aktuellen Zeit (Format wie TIME-Befehl).
`%RANDOM%`	expandiert zu einer Zufallszahl zwischen 0 und 32767.
`%ERRORLEVEL%`	expandiert zum aktuellen ERRORLEVEL-Wert.
`%CMDEXTVERSION%`	expandiert zur Versionsnummer der aktuellen Erweiterungen für den Befehlsinterpreter.
`%CMDCMDLINE%`	expandiert zur ursprünglichen Befehlszeile, die den Befehlsinterpreter aufgerufen hat.
`%HIGHESTNUMANODENUMBER%`	erweitert zu der höchsten NUMA-Knotennummer auf diesem Computer.

Mit `set` können neue Zuweisungen aufgenommen sowie bestehende geändert oder gelöscht werden. Dabei sollten rechts und links vom Gleichheitszeichen keine Leerzeichen stehen. Sethet rechts vom Gleichheitszeichen nichts, wir die Variable gelöscht. Zur Unterscheidung von Programmen schreibt man die Variablennamen oft in Großbuchstaben. Beispiel:

```
@echo off
set BLA=fasel
echo %BLA%
set BLA=
```

Ausgabe:

```
fasel
ECHO ist ausgeschaltet (OFF).
```

Die letzte Meldung rührt daher, dass die Variable BLA nicht definiert ist und daher `echo` ohne Parameter aufgerufen wurde. Beim Abarbeiten der Batch-Datei untersucht DOS jede Zeile vor der Ausführung auf %-Zeichen und ersetzt den Variablennamen durch die zugewiesene Zeichenfolge. Dabei ist es gleichgültig, ob sich die Variable in einem Text, einer Anweisung, einem GOTO-Ziel oder in sonstigen Angaben befindet. Ausgenommen sind nur Labels. Ist eine Variable nicht vorhanden, wird nichts eingesetzt, d. h. der Ausdruck zwischen den %-Zeichen wird einfach entfernt. Um das Prozentzeichen in einem Text zu zeigen, muß es verdoppelt werden.

Wenn der SET-Befehl mit nur einem Variablennamen aufgerufen wird, d.h. ohne Gleichheitszeichen oder einem anderen Wert, wird der Inhalt aller Variablen angezeigt, deren Namen mit eben diesem Buchstaben beginnen. So werden durch

```
SET M
```

alle Variablen angezeigt, die mit dem Buchstaben "M" beginnen. Der SET-Befehl legt den ERRORLEVEL mit 1 fest, wenn der Variablenname nicht in der aktuellen Umgebung gefunden wird.

Bei Variablennamen wird Groß- und Kleinschreibung *nicht* unterschieden. Die zugewiesene Zeichenfolge wird dagegen immer unverändert gespeichert. Variablen-Namen bestehen auch Buchstaben, Ziffern und Sonderzeichen, wobei man sich aber auf den Unterstrich "_" beschränken sollte (obwohl etwas wie `set X+27-3=61` möglich wäre. Sie dürfen jedoch nicht mit einer Ziffer beginnen, denn %0...%9 sind für Befehlsparameter reserviert. Sie sollten auch nicht versehentlich Variablennamen verwenden, die bereits verwendet werden (wie z.B. PATH oder PROMPT) - außer, Sie wollen den Inhalt erweitern, z. B. durch `PATH=%PATH%;C:\dingsda\bin\`.

Windows setzt übrigens einige Variable mit klein geschriebenem Namen, z. B. `windir=C:\Windows`. Damit können diese Variable mit dem `set`-Kommando weder geändert noch gelöscht werden. Mit `set` wird der Variablen der gesamte Rest der Zeile zwischen dem Gleichzeichen und dem Zeilenende zugewiesen. Die Zeichenfolge kann also auch aus mehreren Wörtern, Steuerzeichen usw. bestehen. Nur ein weiteres Gleichheitszeichen in der Zeichenfolge wird bemängelt (Syntax-Fehler). Der Speicherplatz für Umgebungsvariable ist begrenzt. Man sollte daher am Ende einer Batch-Datei alle nicht mehr benötigten Variablen löschen. Das folgende Beispiel zeigt die Verkettung von Strings mit Variablen, den Gebrauch von `shift` und gibt eine Vorschau auf weiter unten beschriebene Batch-Befehle. Der Vergleich `%0X==X` wird genau dann wahr, wenn `%0` keinen Wert enthält und das ist der Fall, wenn alle Parameter bearbeitet wurden.

```
@echo off
:WEITER
if %0X==X goto ENDE
echo Der aktuelle Parameter ist: %0
shift
goto WEITER
:ENDE
echo Fertig ...
```

Ein Aufruf mit drei Parametern ergibt folgende Ausgabe:

```
F:\>t1.bat eins zwei drei
Der aktuelle Parameter ist: t1.bat
Der aktuelle Parameter ist: eins
Der aktuelle Parameter ist: zwei
Der aktuelle Parameter ist: drei
Fertig ...
```

Dem SET-Befehl wurden inzwischen zwei neue Optionen hinzugefügt:

```
SET /A <Ausdruck>
SET /P <Variable>=<Eingabeaufforderungs-Zeichenfolge>
```

Die Option /A legt fest, dass die Zeichenfolge rechts vom Gleichheitszeichen ein numerischer Ausdruck ist, der ausgewertet wird. Das Auswertungsprogramm des Ausdrucks unterstützt dabei die folgenden Operationen, entsprechend ihrer Anordnung mit abnehmendem Vorrang:

```
    ()                  - Gruppierung
    ! ~ -               - unäre Operatoren: Negation, bitweise Invertierung, Vorzeichen
    * / %               - arithmetische Operatoren: Multiplikation, Division, Modulo
    + -                 - arithmetische Operatoren: Addition, Subtraktion
    << >>               - logische Verschiebung links, rechts
    &                   - bitweise UND
    ^                   - bitweise exklusives ODER
    |                   - bitweise ODER
    = *= /= %= += -=    - Verkürzte Schreibweise (wie bei C)
```

```
    &= ^= |= <<= >>=
    ,                   - Trennzeichen für Ausdrücke
```

Bei arithmetischen oder Moduloperatoren müssen Sie die Zeichenfolge für den Ausdruck in Anführungszeichen setzen. Alle nicht-numerischen Zeichenfolgen im Ausdruck werden als Zeichenfolgen von Umgebungsvariablen behandelt, deren Werte vor der Verwendung in Zahlen konvertiert werden. Wenn eine Umgebungsvariable angegeben wird, die nicht definiert ist, wird für diese der Wert Null verwendet. Somit können Sie mit Umgebungsvariablen Berechnungen vornehmen, ohne %-Zeichen einzugeben, um deren Werte zu erhalten.

Wird der Befehl SET /A von der Befehlszeile (außerhalb eines Skripts) ausgeführt, zeigt er den endgültigen Wert des Ausdrucks an. Numerische Werte stellen immer Dezimalzahlen dar, es sei denn, sie haben ein Präfix:
"0x" für hexadezimale Zahlen (z.B. 0x10),
"0b" für binäre Zahlen (z.B. 0b10001010) oder
"0" für oktale Zahlen (z.B. 012).

Die Option /P ermöglicht es, einer Variablen die interaktive Eingabe des Benutzers zuzuweisen. set zeigt die angegebene Eingabeaufforderung an, bevor die Eingabezeile gelesen wird. Die Eingabeaufforderung darf auch leer sein, zum Beispiel:

```
C:\>set /p name=Ihr Name:
Ihr Name: Rembremerdeng

C:\>echo %name%
Rembremerdeng
```

Ablaufsteuerung und `rem`

`rem`
`rem` erlaubt Kommentarzeilen in der Batch-Datei. Hinter `rem` kann ein beliebiger Text stehen. Die Zeile wird nur ausgegeben, wenn ECHO ON eingestellt ist. Damit ist es nicht nur möglich, einfache Kommentare zum besseren Verständnis einzufügen, sondern auch einzelne Befehle, die nicht ausgeführt werden sollen, auszuklammern.

`pause`
Oft ist es nötig, die Abarbeitung der Batch-Datei anzuhalten, um beispielsweise auf das Einlegen einer DC-ROM zu warten. `pause` Unterbricht die Ausführung und gibt den Text "Drücken Sie eine beliebige Taste ..." aus. Wenn man dann eine Taste drückt, wird die Ausführung fortgesetzt (mit "Strg+C" könne man den Ablauf an der Stelle auch abbrechen). Hinter dem Befehl kann ein Text stehen, der dann anstelle des Standardtextes ausgegeben wird.

`timeout`
Das Programm `timeout` verwendet einen Zeitlimitparameter und wartet, bis die angegebene Zeitdauer (in Sekunden) verstreicht oder eine Taste gedrückt wird. Außerdem kann der Befehl den Tastendruck auch ignorieren.

```
timeout /T <Zeitlimit> [/NOBREAK]
```

`/T` <Zeitlimit> Bestimmt die Wartezeit in Sekunden (-1 bis 99999). Ein Wert von -1 bedeutet, dass unendlich lang auf einen Tastendruck gewartet wird. Der Parameter `/NOBREAK` ignoriert gedrückte Tasten, es spielt nur die Zeitangabe eine Rolle. Beispiele:

```
timeout /T 20
timeout /T 360 /NOBREAK
timeout /T -1
```

`choice`
Mit dem Programm `choice` können Benutzer ein Element aus einer Liste auswählen und den Index der Auswahl über den Errorlevel wiedergeben. Damit sind gezielte Eingabe-Abfragen, z. B. Menüs, in einer Batch-Datei möglich. die wichtigsten Parameter sind:

- /C Optionen
  Bestimmt die zu erstellende Auswahlliste. Default ist "JN".
- /CS
  Schaltet die Unterscheidung von Groß-/Kleinschreibung ein. Per Default wird nicht zwischen Groß- und Kleinschreibung unterschieden.
- /M Text
  Meldung, die als Eingabe-Aufforderung angezeigt wird. Ohne Angabe wird nur die Auswahlliste angezeigt.
- /T nnnn /D Auswahl
  Wird innerhalb von nnnn Sekunden keine Taste betätigt, wird die Abarbeitung mit dem durch /D festgelegten Wert (aus der bei /D angegebenen Menge) fortgesetzt.

Die Umgebungsvariable ERRORLEVEL wird auf den Index des Wertes gesetzt, der aus der Liste ausgewählt wurde. Das erste Zeichen ergibt den Wert 1, die zweite den Wert 2 usw. Falls der Benutzer eine unzulässige Taste drückt, erfolgt ein akustisches Signal. Im Fehlerfall wird ein ERRORLEVEL von 255 zurückgegeben. Beim Drücken von [STRG]+[Untbr] oder [STRG]+[C] wird ein ERRORLEVEL-Wert von 0 wiedergegeben. Beispiele:

```
choice /C JNA /M "Drücke J für Ja, N für Nein oder A für Abbrechen"
choice /C XYZ /T 10 /D Y
choice /C 1234 /M "Wählen Sie die gewünschte Option "
```

ERRORLEVEL
`ERRORLEVEL` ist keine "richtige" Variable, sondern eher ein Kommando, das einen Wert liefert. so erlaubt es DOS nicht, den Errorlevel in Form einer Variablen zu verwenden. Fast jedes DOS-Programm liefert beim Beenden einen Abschluß-Fehlercode. Falls das Programm fehlerfrei ablief, so gibt es in der Regel den Wert 0 zurück, falls es Fehler gab eine Fehlernummer. Diese Fehlernummer ist der sogenannte `ERRORLEVEL`. Sie kann aber nur mittels `if`-Befehl abgefragt werden. Wichtig ist, dass hier immer mit der größten Nummer angefangen wird, weil der `if`-Befehl nicht auf Gleichheit, sondern auf größer oder gleich prüft:

- `if ERRORLEVEL n` bedeutet: "*if [Return-Code]* >= *n*"

Zur Speicherung des ERRORLEVEL in einer "echten" Variablen gibt es auf den ersten Blick nur die Methode mit 255 einzelnen `if`Abfragen und `set`-Befehlen:

```
if ERRORLEVEL 255 set ERROR=255
if ERRORLEVEL 254 set ERROR=254
if ERRORLEVEL 253 set ERROR=253
   ...
if ERRORLEVEL 3 set ERROR=3
if ERRORLEVEL 2 set ERROR=2
if ERRORLEVEL 1 set ERROR=1
if ERRORLEVEL 0 set ERROR=0
```

Kontrollstrukturen

Kontrollstrukturen bieten die Möglichkeit, den linearen Ablauf einer Batch-Datei zu verändern. Dazu stehen hier leider nur wenige und unstrukturierte Befehle zur Verfügung, es reicht jedoch für die wichtigsten Aufgaben.

goto
Mit dem Befehl `goto` können Sie den Kommandointerpreter veranlassen, nicht in der nächsten Zeile mit der Bearbeitung fortzufahren, sondern zu einer bestimmten Stelle in der Batch-Datei zu springen. Diese Stelle wird durch eine Marke gekennzeichnet, die aus einem Markennamen mit nachfolgendem Doppelpunkt besteht. So lassen sich z. B. Schleifen programmieren:

```
@echo off
:WEITER
echo Hallo Welt
```

```
shift
goto WEITER
```

Aus so einer "unendlichen Schleife" kommt man nur durch Drücken von [Strg]-[Untbr] oder [Strg]-[C] wieder heraus.

`if`
`if` dient der Abfrage von Bedingungen. Sie können drei verschiedene Formen von Bedingungen überprüfen, die Existenz einer Datei, die Gleichheit zweier Zeichenketten und den Rückgabewert (ERRORLEVEL) des zuletzt gelaufenen DOS-Programms. Einziger Operator ist `NOT`, um die Bedingung umzukehren. Es gibt keinerlei logische Verknüpfungen. Nach der Bedingung steht auch ein einziger Befehl oder Kommandoaufruf. Deshalb werden in der Regel `if`-Befehle mit dem `goto` verknüpft, wenn abhängig von der Bedingung mehrere Befehle ausgeführt werden sollen.

Das Format des Befehls ist `if <Bedingung> <Befehl>`. Als Bedingung kommen folgende Ausdrücke in Frage:

- `exist <Dateiname>`
  Abfrage, ob eine bestimmte Datei vorhanden ist. Umgekehrt kann mit `not exist` das Fehlen einer Datei festgestellt werden. Um auch die Existenz eines Verzeichnisses festzustellen, verwendet man einen Trick. In jedem existierenden Verzeichnis kann man das Pseudo-Gerät NUL als Datei finden. Gibt es das Verzeichnis nicht, dann wird auch die "Datei" NUL nicht gefunden:

  ```
  if not exist C:/Daten/NUL mkdir Daten
  ```

- `string1==string2`
  prüft die Gleichheit von zwei Zeichenketten. Solange eine Variable nur ein einziges Wort enthält, sind `if`-Abfragen problemlos. Aber auch hier gelten die Grundregeln, dass Groß- und Kleinschreibung unterschieden wird und leere Variable die Syntax gefährden. Zur Abfrage auf leere Variable bzw. von möglicherweise leeren gab es oben schon ein Beispiel für einen leeren Parameter. Bei Variablen bieten eckige Klammern oder Gänsefüsschen einen optisch befriedigenden Ansatz:

  ```
  if "%LW%"=="C:" goto WEITER                Laufwerksangabe vorhanden
  if [%LW%]==[]   goto FEHLER                Laufwerksangabe fehlt
  ```

  Wenn eine Variable aus mehrere Wörtern, getrennt durch Leerzeichen, Komma, Semikolon etc. besteht, können Probleme entstehen. Mehrere Wörter *nach* dem "==" führen zu Syntaxfehlern (verglichen wird ohnehin nur das erste Wort). Bei mehreren Wörtern *vor* dem "==" erolgt zwar keine Fehlermeldung, es wird jedoch nur das erste Wort verglichen.
- `ERRORLEVEL nn`
  vergleicht den Inhalt der ERRORLEVEL-Variablen (siehe oben) mit der nachfolgenden Zahl nn. Denken Sie daran, dass die ERRORLEVEL-Abfrage eigentlich eine ">="-Bedingung ist. Auch hier kann durch ein vorangestelltes `not` die Bedingung negiert werden.

Beispiel:

```
@echo off
:WEITER
CHOICE /C:ABCE /M "Drücken sie eine der Tasten"
if ERRORLEVEL 4 goto ENDE
if ERRORLEVEL 3 echo C eingegeben.
if ERRORLEVEL 2 echo B eingegeben.
if ERRORLEVEL 1 echo A eingegeben.
goto WEITER
:ENDE
echo Uffff....
```

Ein Aufruf ergibt etwa folgende Ausgabe:

```
Drücken sie eine der Tasten [A,B,C,E]?A
A eingegeben.
Drücken sie eine der Tasten [A,B,C,E]?C
C eingegeben.
B eingegeben.
```

```
A eingegeben.
Drücken sie eine der Tasten [A,B,C,E]?B
B eingegeben.
A eingegeben.
Drücken sie eine der Tasten [A,B,C,E]?E
Uffff....
```

Sie sehen schon, dass es nicht ganz so wie gedacht läuft. Je nach Eingabe werden mehere Echo-Zeilen ausgegeben. Also muss die Batch-Datei noch weiter entwickelt werden:

```
@echo off
:WEITER
CHOICE /C:ABCE /M "Drücken sie eine der Tasten"
if ERRORLEVEL 4 goto ENDE
if ERRORLEVEL 3 goto C
if ERRORLEVEL 2 goto B
if ERRORLEVEL 1 goto A
:C
echo C eingegeben.
goto WEITER
:B
echo B eingegeben.
goto WEITER
:A
echo A eingegeben.
goto WEITER
:ENDE
echo Uffff....
```

Nun stimmt auch die Ausgabe:

```
Drücken sie eine der Tasten [A,B,C,E]?A
A eingegeben.
Drücken sie eine der Tasten [A,B,C,E]?B
B eingegeben.
Drücken sie eine der Tasten [A,B,C,E]?C
C eingegeben.
Drücken sie eine der Tasten [A,B,C,E]?E
Uffff....
```

Neben dem alten Gleichheitsoperator "==" und der Negation durch NOT, die beide noch vorhanden sind, bietet CMD.exe nun einen vollständigen Satz von Vergleichsoperatoren:
br>

`EQU`	Gleichheit (synonym für "==")
`NEQ`	Ungleichheit
`LSS`	kleiner
`LEQ`	kleiner oder gleich
`GTR`	größer
`GEQ`	größer oder gleich

Leider vermeiden die Operatoren jede Ähnlichkeit mit anderen Script-Sprachen. Sie erlauben den Vergleich von numerischen Werten und auch von Zeichenketten. Dabei ignoriert der Interpreter sogar Groß- und Kleinschreibung, sofern man "`if`" mit der Option /i verwendet.

Achtung: Beim Vergleich von Zeichenketten, wertet CMD.exe hier Großbuchstaben größer als Kleinbuchstaben:

```
C:\>if "GROSS" GTR "gross" echo Jau
Jau
```

CMD.exe läßt in "IF"-Anweisungen auch einen "ELSE"-Zweig zu. Will man etwa die IF/ELSE-Abfrage in einer Zeile formulieren, dann muss der von "IF" abhängige Befehl in Klammern gesetzt werden. Achten Sie

darauf, dass vor der Klammer immer ein Leerzeichen stehen muss:

```
C:\>if 8 GTR 7 (echo 8) else (echo 7)

C:\>set X=8
C:\>if "%X%" GTR "7" (echo %X%) else (echo 7)
8
```

Auf die klassische Weise der Programmierung wäre das länger geraten:

```
if "%X%" GTR "7" goto MARKE1
echo 7
goto MARKE2
:MARKE1
echo %X%
:MARKE2
rem und hier dann weiter im Script
```

Die neue strukturierte if-Anweisung funktioniert auch mit mehreren Befehlen und mehrzeilig, z. B.:

```
if <Bedingung> (
   Befehl1
   Befehl2
 ) else (
   Befehl3
   Befehl4
 )
```

Beispiel: Überprüfen, ob ein bestimmtes Programm schon läuft:

```
@echo off
set Programm=%1

tasklist | find /i "%Programm%" >nul

if %errorlevel% == 0 (
  echo Das Programm "%Programm%" laeuft gerade!
) else (
  echo Das Programm "%Programm%" laeuft nicht!
)
```

Einige Beispielaufrufe:

```
C:\>prog.bat explorer
Das Programm "explorer" laeuft gerade!

C:\>prog.bat notepad
Das Programm "notepad" laeuft gerade!

C:\>prog.bat nixda
Das Programm "nixda" laeuft nicht!
```

for

Führt einen Befehl für jeden einzelnen Wert einer Liste aus. Diese Liste steht in Klammern und kann aus einzelnen Zahlen oder Worten bestehen, aber auch eine oder mehrere Dateinamen mit Jokerzeichern enthalten ("?" steht für ein beliebiges Zeichen und "*" für eine Folge beliebiger Zeichen). Die Syntax des Befehls lautet:

```
for %%Laufvariable in (Liste) do <Kommando>
```

Im Gegensatz zu den an anderer Stelle verwendeten Variablen, wird die Laufvariable nicht in %...% eingeschlossen, sondern hat zwei Prozentzeichen vor dem Namen. Wenn sie im Kommando als Parameter verwendet wird, muss auch hier "%%" verwendet werden. Beispiele:

```
for %%d in (*.TXT) do copy %%d D:\Texte\
for %%1 in (dat1 dat2 dat3) do del %%1
for %%d in (*.*) echo %%d
```

Die doppelten Prozentzeichen sind wegen der recht rudimentären Variablenbehandlung des Kommandointerpreters nötig. Wenn Sie `for` direkt auf der Kommandozeile anwenden reicht nämlich ein Prozentzeichen. Statt einer Liste darf auch eine Variable in der Klammer stehen. Diese Variable kann auch eine ganze Pararmeter- oder Argument-Liste enthalten, die sich dann mit `for` wieder zerlegen läßt. Beispiel:

```
for %%a in (%PATH%) do echo %%a
```

Das zeigt auch, dass nicht nur das Leerzeichen Trenner für die Listenelemente ist, sondern auch ";" oder "," verwendet werden können. Das folgende Beispiel zeigt die Kombination von `for` mit dem im folgenden Abschnitt beschriebenen `call`-Befehl:

```
@ECHO OFF
REM Gibt die Zahlen 00 - 99 aus
REM Diese Datei muss count.bat heißen

if "%1" == ""       goto outer
if "%1" == "output" goto output
if "%1" == "inner"  goto inner
goto exit

:inner
for %%i IN (0 1 2 3 4 5 6 7 8 9) do call count output %2 %%i
goto exit

:outer
for %%i IN (0 1 2 3 4 5 6 7 8 9) do call count inner %%i
goto exit

:output
echo %2%3

:exit
```

Wenn die Befehlserweiterungen aktiviert sind, werden folgende Ergänzungen für den for-Befehl unterstützt:

- `for /D %Variable in (Satz) do <Kommando>`
  Enthält der Satz Jokerzeichen, dann bezieht sich der for-Befehl auf Verzeichnisse und nicht auf Dateien, z. B.:

  ```
  for /D %%N in (*.*) do echo %%N
  ```

- `for /R <[[Laufwerk:]Pfad]> %Variable in (Satz) do <Kommando>`
  Der for-Befehl wird ausgehend vom angegebenen Verzeichnis [Laufwerk:]Pfad für jedes darunterliegende Verzeichnis ausgeführt. Wird kein Verzeichnis nach der /R-Option angegeben, verwendet for das aktuelle Verzeichnis. Wenn der Satz nur einen einzelnen Punkt als Zeichen (.) enthält, wird nur die Verzeichnisstruktur aufgelistet.
- `for /L %Variable in (Start,Schritt,Ende) do <Kommando>`
  Der Satz ist eine Folge von Zahlen von Start bis Ende und der angegebenen Schrittweite. So erstellt (1,1,7) die Folge 1 2 3 4 5 6 7 und (9,-1,4) die Folge (9 8 7 6 5 4).

Der for-Befehl erlaubt nun auch, wie "if" eine strukturierte Form, bei der die Anweisungen nach dem "do" geklammert werden und mehrzeilig geschrieben werden können, z. B.:

```
for /D %%N in (*.*) do (
  cls
  @echo %%N
)
```

call

Aufruf und Ausführen einer anderen Batchdatei. Nach deren Abarbeitung wird in der auf den `call`_Befehl folgenden Zeile fortgefahren. In einer Batch-Datei lassen sich nicht nur ausführbare Programme aufrufen, sondern auch andere Batch-Dateien. Verwendet man nur den Namen einer Batch-Datei als Kommando, wird die Ausführung in der aufgerufenen Batch-Datei fortgesetzt, aber nicht zur ursprünglichen Batch-Datei

zurückgekehrt. Erst mit dem `call`-Befehl wird eine Batch-Date wie ein ausführbaren Programm behandelt. Es gilt demnach:

- `XYZ.BAT` setzt den Ablauf in XYZ.BAT fort, **ohne** zum aufrufenden Batch zurückzukehren,
- `call XYZ.BAT` kehrt nach Ausführung von XYZ.BAT zurück.

Sollen vom aufgerufenen Batch-Job Werte zurückgegeben werden, sind dafür Umgebungsvariable zu verwenden. Der zuletzt von einem Programm erzeugte Return-Code kann noch nach der Rückkehr zum aufrufenden Batch-Job per ERRORLEVEL abgefragt werden. Die Batch-Datei selbst erzeugt keinen Return-Code.

`pushd/popd`
`pushd` wechselt zum angegebenen Pfad und speichert den aktuellen Pfad bis zum Aufruf von `popd`. `popd` wechselt zurück zum gespeicherten Pfad. Die Befehle können geschachtelt werden. Beispiel:

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

Das `start`-Kommando

Startet ein Programm in einem neuen eigenen Windows-Fenster. Auf diese Weise kann man aus der Batch-Datei heraus auch Windows-Anwendungen starten. Die Syntax des Kommandos lautet:

```
START "Titel" /D <Startverzeichnis> <Optionen> Programm
```

Folgende (optionale) Optionen sind möglich:

- `"Titel"`: Der Titel des neuen Fensters
- `/D <Startverzeichnis>`: Das Startverzeichnis des aufgerufenen Programms
- `/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].

Beispiel: Windows öffnet ein neues Fenstern mit Eingabeaufforderung:

```
start "Ich bin das neue Fensters" /WAIT /NORMAL cmd.exe
```

Es gilt dabei: `/WAIT` = Wartet bis Anwendung geschlossen wird, `/NORMAL` = Startet Anwendung in der NORMAL-Prioritätsklasse. Man kann aber auch den Browser starten und in der Batch-Datei weitermachen:

```
start "Ich bin das neue Fensters" /D C:\temp /NORMAL firefox.exe
```

Im Prinzip kann man das mit jedem Programm machen. Ebenso lassen sich beliebige Parameter übergeben. Nützliche (System)-Programme für Batch-Abläufe sind u. a.:

regedit.exe = Registrierungseditor
explorer.exe = Windows Ordner Explorer
taskmgr.exe = Windows Taskmanager

taskeng.exe = Aufgabenplanungsmodul
calc.exe = Taschenrechner
firefox.exe = Firefox-Internetbrowser
notepad.exe = Standard-Textbearbeitungsprogramm
cmd.exe = Kommandointerpreter für Batch

Letzte Aktualisierung: 09/09/2017 11:52:33