

M122

Windows Kommandointerpreter und Batch



Eine Anleitung



Box of punch cards containing several computer programs.

*Diese Unterrichtsunterlage basiert auf den Unterlagen des
Berufsbildungszentrums Sursee M122_Sursee.pdf, 2005.*

Überarbeitet und ergänzt durch P. Rutschmann BBW

Inhaltsverzeichnis

Inhaltsverzeichnis	2
1. Wozu ein Kommandointerpreter?	4
2. DOS Batch - Programmierung	5
2.1 Grundlagen	5
2.1.1 Starten des Kommandointerpreters	5
2.1.2 Symbolik in der Hilfe zum Programm	6
2.1.3 Zusätzliche Hilfe	7
2.1.4 Built-ins versus Utilities	7
3. Stapelverarbeitung (Batch-processing)	8
3.1 Aufruf	8
3.2 Manueller Abbruch	9
3.3 Automatischer Unterbruch mit PAUSE	9
3.4 Ausgabe von Texten mit ECHO	9
3.4.1 Text Ausgabe	9
3.4.2 Unterdrücken der Ausgabe der Befehle.	9
3.4.3 Ausgabe einer Leerzeile	10
3.5 Kommentare in der Stapeldatei	10
3.6 Spezielle Zeichen	10
3.6.1 Platzhalter (Wildcards)	10
3.6.2 Double-Quotes	10
3.6.3 Escaped characters	10
3.7 Kontrollstrukturen	11
3.7.1 Sprung an eine bestimmte Stelle	11
3.7.2 Verzweigung mittels IF	11
3.7.3 Wiederholung mittels FOR	14
3.7.4 Auswahl mittels CHOICE	15
3.8 Kommandozeilen Parameter	16
3.8.1 Mehr als 9 Parameter	16
3.9 Variablen	17
3.9.1 Setzen und Löschen von Umgebungsvariablen	17
3.9.2 Verwendung von Umgebungsvariablen	18
3.10 Starten und Aufrufen	18
3.10.1 Einen neuen Kommandointerpreter starten	18
3.10.2 Ein neues Kommandointerpreter Fenster starten	18

3.10.3	Eine andere Batch-Datei starten	19
3.11	Ausgaben umleiten.....	20
3.12	Weitere Operatoren	21
4.	Übersicht über die Befehle von cmd.....	22

1. Wozu ein Kommandointerpreter?

Die ersten Personal Computer hatten nur einfache Bedienungsprogramme, sogenannte **Kommandointerpreter**. Diese haben die Aufgabe, Eingaben des Benutzers in Aktionen umzusetzen und andere Programme zu starten. Diese Kommandointerpreter sind selbst kleine Programme, welche beim Starten des Systems von einem sogenannten Boot-Loader (Startlader) automatisch in den Speicher geladen und gestartet wurden.

Unter Unix haben diese Interpreter Namen wie *C-Shell*, *Korn-Shell*, etc. erhalten.

Viele Kleinststeuerungen, welche häufig nicht über eine grafische Benutzeroberfläche verfügen, warten nach dem Start in ihrem Kommandointerpreter auf Benutzereingaben oder starten direkt aus ihrem Kommandointerpreter heraus das Programm, das ihrem Verwendungszweck dient.

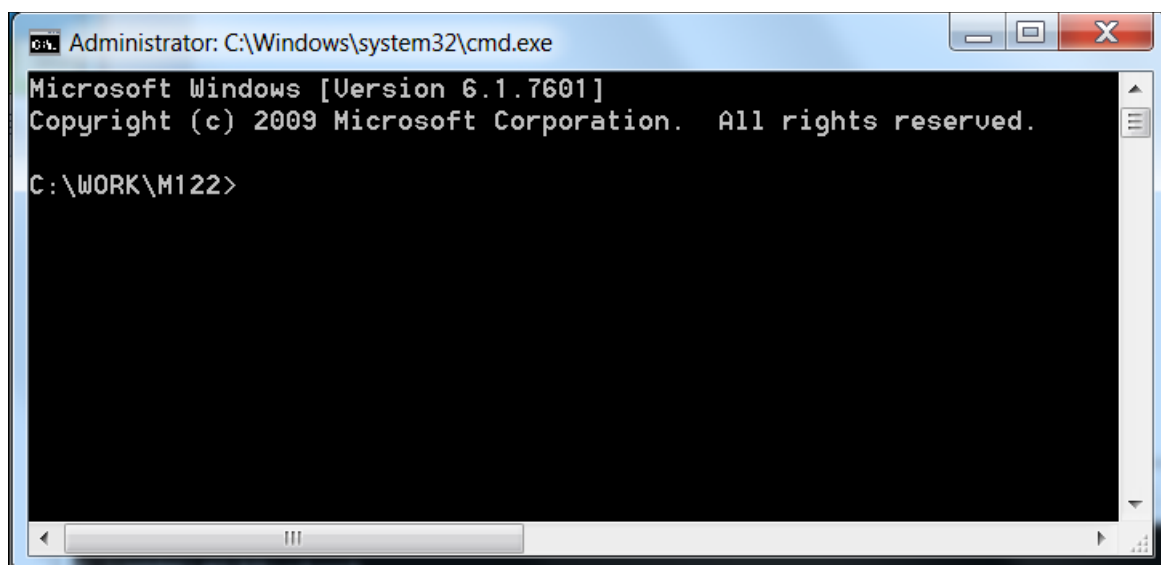
In diesem Dokument wird der Kommandointerpreter von Microsoft vorgestellt. Dieser hat den Namen COMMAND.COM (DOS, Win95/98/ME) respektive CMD.EXE (NT-Systeme).

Rasch war das Bedürfnis da, auf einen Befehl nicht nur eine, sondern eine ganze Reihe vom Benutzer definierte Aktionen auszuführen. Die Aufgaben resp. Kommandos werden dabei in einfache Text- Dateien geschrieben, welche anstelle des Benutzers, die Eingaben für den Kommandointerpreter liefern. Diese Programme werden **BATCH-Programme oder BATCH-Scripts** genannt und haben unter Windows die Endung *.BAT*.

Der Name kommt aus den Zeiten der Grossrechner und Lochkarten. Auf unzähligen Karten ist im Muster der Löcher das Programm für den Rechner gestanzt. Um ein Programm zu laden, musste der Grossrechner mit diesem Kartenstapel gefüttert werden, woraus sich der Begriff Stapelverarbeitung (engl. batch processing) ableitet.

Mit Einführung von Windows NT hat ein weiterer Kommandointerpreter Einzug gehalten, welcher den kürzeren Namen *CMD.EXE* erhalten hat. Er hat gegenüber COMMAND.COM einige Erweiterungen erhalten. Heutige Windows-Systeme stellen beide Interpreter und darüber hinaus noch den Kommandointerpreter Power-Shell zu Verfügung.

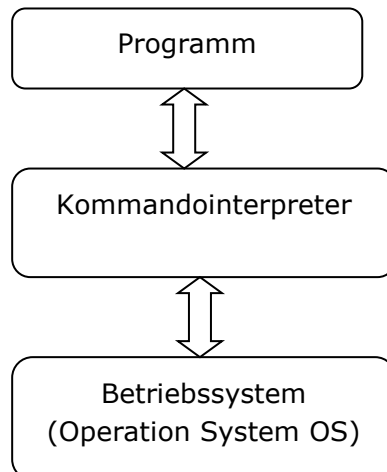
Falls Sie einmal einen alten DOS-Rechner in die Hände bekommen (siehe Grossvaters Estrich©), werden Sie es mit dem COMMAND.COM aufnehmen müssen.



2. DOS Batch - Programmierung

2.1 Grundlagen

Ein Kommandointerpreter ist ein Programm, das im Kontext (Umfeld) des Betriebssystems läuft. Im Kommandointerpreter können Kommandos ausgeführt und weitere Programme gestartet werden.



2.1.1 Starten des Kommandointerpreters

Zum Öffnen des Kommandointerpreters geben sie unter „START/Ausführen als“ den Befehl COMMAND bzw. CMD ein. Mit der Tastenkombination ALT-ENTER können sie zwischen Fenster- und Vollbild-Ansicht umschalten.

Wie viele Programme, verfügt der Kommandointerpreter über eine eingebaute Hilfe, welche zeigt wie das Programm aufgerufen werden kann.

Dazu wird das Programm mit dem Parameter `/?` aufgerufen:

COMMAND.COM:

```
C:\>COMMAND /?
```

Startet eine neue Kopie des MS-DOS-Befehlsinterpreters.

```
COMMAND [[Laufwerk:]Pfad]] [Gerät] [/E:nnnnn] [/P] [/C Befehl] [/MSG]
```

[Laufwerk:]	Pfad Bezeichnet das Verzeichnis mit der Datei COMMAND.COM.Gerät Gerät für die Ein- und Ausgabe des Befehlsprozessors.
/E:nnnnn	Stellt die anfängliche Umgebungsgrösse auf nnnnn Bytes ein.
/P	Macht den neuen Befehlsinterpreter permanent (nicht beendbar) .
/C	Befehl Führt den Befehl in Zeichenkette aus und endet dann.
/MSG	Alle Fehlermeldungen werden im Arbeitsspeicher gehalten (nur zusammen mit der Option /P verwendbar) .

CMD.EXE

C:\>CMD /?

Starts a new instance of the Windows command interpreter

CMD [/A | /U] [/Q] [/D] [/E:ON | /E:OFF] [/F:ON | /F:OFF] [/V:ON | /V:OFF]
[[/S] [/C | /K] string]

/C Carries out the command specified by string and then terminates
/K Carries out the command specified by string but remains
/S Modifies the treatment of string after /C or /K (see below)
/Q Turns echo off
/D Disable execution of AutoRun commands from registry (see below)
/A Causes the output of internal commands to a pipe or file to be ANSI
/U Causes the output of internal commands to a pipe or file to be
Unicode
/T:fg Sets the foreground/background colors (see COLOR /? for more info)
/E:ON Enable command extensions (see below)
/E:OFF Disable command extensions (see below)
/F:ON Enable file and directory name completion characters (see below)
/F:OFF Disable file and directory name completion characters (see below)
/V:ON Enable delayed environment variable expansion using ! as the
delimiter. For example, /V:ON would allow !var! to expand the
variable var at execution time. The var syntax expands variables
at input time, which is quite a different thing when inside of a FOR
loop.
/V:OFF Disable delayed environment expansion.

2.1.2 Symbolik in der Hilfe zum Programm

Name	Erklärung	Beispiel
Klammer: []	Abgrenzung eines zusammengehörenden Teils (Block). Die einzelne Blöcke sind optional; d.h. sie können, müssen aber nicht, angewandt werden. Die Klammern [] werden beim Aufruf des Programmes weggelassen.	[[Laufwerk:]Pfad]]
Entweder oder: 	Unterscheidung von verschiedenen Möglichkeiten für Blöcke. Entweder dieser oder der andere Block kann angewandt werden.	[/F:ON /F:OFF]
Option	Zur Auswahl stehende Möglichkeiten, mit denen das Verhalten des Programms gesteuert wird.	/A
Parameter	Eingabewert mit dem das Programm arbeiten wird.	Verzeichnis

2.1.3 Zusätzliche Hilfe

Die Kommandointerpreter bietet eine weitere Hilfe an, welche mit *HELP* aufgerufen werden kann. Diese Funktionalität ist umso wichtiger, als mit jeder neuen Betriebssystem-Version Ergänzungen und Änderungen in der Funktionalität und Lieferumfang gemacht werden kann. Daher sollten Programme, bevor sie ausgeliefert werden, auf den jeweiligen Betriebssystemen getestet werden.

```
C:\Users\rut>help
```

For more information on a specific command, type HELP command-name

```
ASSOC          Displays or modifies file extension associations.
ATTRIB         Displays or changes file attributes.
BREAK          Sets or clears extended CTRL+C checking.
BCDEDIT        Sets properties in boot database to control boot loading.
CACLS          Displays or modifies access control lists (ACLs) of files.
CALL           Calls one batch program from another.
CD             Displays the name of or changes the current directory.
```

... (Aufzählung nicht vollständig wiedergegeben.)

2.1.4 Built-ins versus Utilities

Im Kontext des Kommandointerpreters können verschiedenste Befehle einzeln, hintereinander oder miteinander verkettete aufgerufen werden. Wie beim Betrachten der Ausgabe der Hilfe gesehen, steht jeder Befehl für eine spezifische Aufgabe. ZBsp *DIR*: Ausgabe des Inhaltes eines Verzeichnisses.

Bei der Eingabe eines Befehls wird Gross-Kleinschreibung nicht unterschieden: zBsp. sind *DIR*, *dir* oder *Dir* möglich.

Viele der grundlegenden Befehle wie *dir*, *cd*, etc. sind im Kommandointerpreter direkt implementiert. (Built-In)

Neuere Erweiterungen werden als externe Programme realisiert (Utilities) und müssen im Such-Pfad sein, damit sie gefunden werden (Bsp. CHOICE.EXE siehe weiter unten).

Was alles zum aktuellen Such-Pfad gehört, kann mittels *PATH* ausgegeben werden.

```
C:\WORK\M122>PATH
PATH=C:\Program Files (x86)\PC Connectivity Solution
.0\;C:\Program Files\TortoiseSUN\bin;C:\Program File
k\sys\x64\;C:\Program Files (x86)\Java\jre7\bin;
```

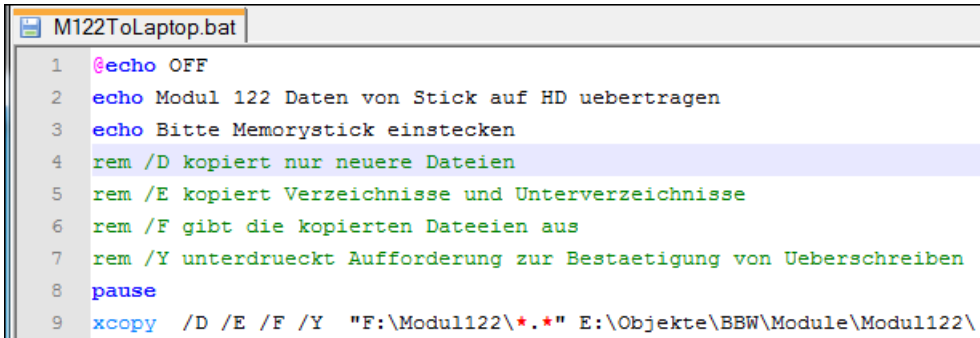
Auszug aus PATH

3. Stapelverarbeitung (Batch-Processing)

Wie schon erwähnt, können die vom Kommandointerpreter bearbeitbaren Befehle auch in einer Textdatei zu einem Ablauf zusammengefasst werden. Zum einen vereinfacht dies die wiederholte Anwendung, zum anderen den Umgang mit den nicht immer einfach anwendbare Optionen und Parameter.

Diese Batch Datei trägt die Endung *.bat*. Sie ist eine reine Textdateien (ASCII) und können mit jedem beliebigen Editor (z.B. EDIT) erstellt werden.

Beispiel:



```
M122ToLaptop.bat
1 @echo OFF
2 echo Modul 122 Daten von Stick auf HD uebertragen
3 echo Bitte Memorystick einstecken
4 rem /D kopiert nur neuere Dateien
5 rem /E kopiert Verzeichnisse und Unterverzeichnisse
6 rem /F gibt die kopierten Dateien aus
7 rem /Y unterdrueckt Aufforderung zur Bestaetigung von Ueberschreiben
8 pause
9 xcopy /D /E /F /Y "F:\Modul122\*.*)" E:\Objekte\BBW\Module\Modul122\
```

3.1 Aufruf

Der Name der aufzurufenden Stapeldatei kann **direkt im Fenster des Kommandointerpreters** eingegeben und mit *Enter* gestartet werden.

Der Vorteil dabei ist, dass man allfällige Ausgaben der Stapeldatei im Fenster des Kommandointerpreters auch noch sieht, wenn die Verarbeitung abgeschlossen ist.



```
C:\Users\peter.rutschmann>M122ToLaptop.bat
Modul 122 Daten von Stick auf HD uebertragen
Bitte Memorystick einstecken
Press any key to continue . . .
F:\Modul122\04_Aufgaben\03_WindowsScripts\Rutschmann_Peter\9
Rutschmann_Peter\90_Script\M122ToLaptop.bat
1 File(s) copied
C:\Users\peter.rutschmann>
```

Man kann die Stapeldatei auch im Explorer **mittels Doppelklick** starten. Es wird sich für die Dauer der Ausführung ein Fenster mit einem Kommandointerpreter öffnen. Da sich dieses nach der Ausführung sofort wieder schliesst, wird man Mühe haben, die Ausgaben der Stapeldatei lesen zu können. Während der Entwicklung einer Stapeldatei ist diese Art der Ausführung deshalb nicht zu empfehlen.

3.2 Manueller Abbruch

Will man die Ausführung abbrechen, so kann man während der Ausführung die Tastenkombination *Ctrl+C* drücken. Und die Frage erscheinende Frage mit *J* resp. *Y* beantworten.

```
C:\Users\peter.rutschmann>M122ToLaptop.bat
Modul 122 Daten von Stick auf HD uebertragen
Bitte Memorystick einstecken
Press any key to continue . . .
Terminate batch job (Y/N)? y
C:\Users\peter.rutschmann>
```

Diese Möglichkeit ist äusserst nützlich, wenn die Stapeldatei während der Entwicklung in einer Schleife hängen bleibt.

3.3 Automatischer Unterbruch mit PAUSE

Oft ist es nötig, die Abarbeitung der Stapeldatei an einer bestimmten Stelle anzuhalten, um z.B. auf das Einstecken des Memorysticks zu warten.

Dazu kann man den Befehl *PAUSE* benützen.

Der Ablauf wird unterbrochen und eine Aufforderung, eine Taste zu drücken, wird ausgegeben. (Zeile 8 im Beispiel oben)

```
C:\Users\peter.rutschmann>M122ToLaptop.bat
Modul 122 Daten von Stick auf HD uebertragen
Bitte Memorystick einstecken
Press any key to continue . . .
```

3.4 Ausgabe von Texten mit ECHO

Der Befehl *ECHO* hat verschiedene Wirkungen.

3.4.1 Text Ausgabe

Der Befehl *ECHO* wird benutzt, um beliebige Textausgaben auf dem Bildschirm erscheinen zu lassen. Dabei wird alles, was hinter einem *ECHO* Befehl steht als Textzeile auf dem Schirm abgebildet.

(Zeile 2 und 3 im Beispiel oben)

3.4.2 Unterdrücken der Ausgabe der Befehle.

Wenn eine Batch-Datei abgearbeitet wird, so werden die darin enthaltenen Befehle auf dem Bildschirm angezeigt.

Diese Eigenschaft ist oft störend und kann daher aus- und auch wieder eingeschaltet werden. Das geschieht mit dem Befehl *ECHO OFF*.

Damit der Befehl *ECHO OFF* auch nicht dargestellt wird, wird ihm ein Klammeraffe (@) vorgestellt. (Zeile 1 im Beispiel oben)

Üblicherweise wird das Echo in Batch-Dateien gleich zu Anfang abgeschaltet.

Ohne jeden Parameter gibt *ECHO* den jeweiligen Zustand (ON oder OFF) aus.

3.4.3 Ausgabe einer Leerzeile

Um eine Leerzeile auszugeben müssen Sie den *ECHO*-Befehl mit direkt folgendem Punkt angeben.

3.5 Kommentare in der Stapeldatei

Innerhalb einer Batch-Datei werden Kommentare mit dem Befehl *REM* eingeleitet. Zeilen, die mit diesem Befehl beginnen, werden vom Kommandointerpreter einfach ignoriert. Damit ist es nicht nur möglich, einfache Kommentare zum besseren Verständnis einzufügen, sondern auch einzelne Befehle, die nicht ausgeführt werden sollen auszuklammern.

(Zeile 4 bis 7 im Beispiel oben)

Alternativ können am Anfang einer Zeile auch zwei Doppelpunkte stehen (:). Solche Zeilen gelten auch als Kommentare.

3.6 Spezielle Zeichen

3.6.1 Platzhalter (Wildcards)

Wildcards sind Platzhalter für andere Zeichen. Auch der Kommandointerpreter kennt solche Wildcards:

- Das **?** steht für ein **einzelnes** Zeichen.
So steht das ? in M?ier.doc für eine beliebigen Zeichen. Man findet alle Dokumente wie Meier.doc, Maier.doc usw.
- Der ***** (Asterix) steht für eine beliebige Anzahl beliebiger Zeichen.
So steht das * in *.doc für eine beliebige Anzahl beliebiger Zeichen. Man findet alle Dokumente mit der Endung .doc.

3.6.2 Double-Quotes

Normalerweise wird ein Leerzeichen bei der Eingabe eines Befehls, als Trenner zwischen Optionen und/oder Parametern verstanden. Gibt man als Parameter einen (Text) mit einem Leerzeichen ein, so kann wird das Leerzeichen als Trennung interpretiert, der Parameter damit falsch verstanden.. wir haben ein Problem!!

Mittels Double Quotes, gemeint sind doppelte Anführungsstriche "", kann man einen String (Text) als Einheit markieren.

Ein mit Quotes umfasster Text Beginnt hinter dem ersten " und endet vor dem zweiten ".

```
"Dieser Text wird als Ganzes genommen."
```

Leerzeichen werden nicht als Trennung interpretiert, Problem beseitigt☺.

3.6.3 Escaped characters

Will man reservierte Zeichen als Teil der Argumente für einen Befehl verwenden, so muss man verhindern, dass der Kommandointerpreter die Zeichen selber interpretiert.

```
C:\>echo <dir>
```

Dies funktioniert nicht richtig. Da die <> Zeichen vom Kommandointerpreter falsch verstanden werden.

Lösung Hilfe von ^:

```
C:\>echo ^<dir^>
```

So wird <dir> ausgegeben.

Die Benützung von ^ ist selten und kommt nur in sehr speziellen Situationen zum Tragen.

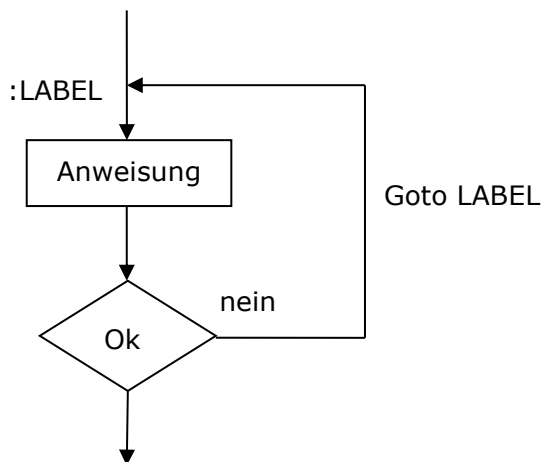
3.7 Kontrollstrukturen

Kontrollstrukturen sind Möglichkeiten, den Ablauf einer Batch-Datei zu verändern.

3.7.1 Sprung an eine bestimmte Stelle

Mit dem Befehl `GOTO` können Sie den Kommandointerpreter dazu bringen, nicht in der nächsten Zeile mit der Bearbeitung der Datei fortzufahren, sondern sie befehlen ihm zu einer bestimmten Stelle in der Datei zu springen. Diese Stelle wird als Marke (Label) bezeichnet und mit einem Doppelpunkt und einem Namen definiert.

Diese Sprünge ermöglichen Wiederholungen oder auch das Überspringen von Abschnitten im in der Stapeldatei.



```
IF_GOTO_LABEL.bat
1  :LABEL
2
3  ECHO Bitte geben Sie OK ein
4  set /P Eingabe=
5
6  IF NOT "%EINGABE%"=="OK" GOTO LABEL
7
8  ECHO ENDE
```

WARNUNG:

Die Möglichkeit, in einem Ablauf beliebig zu springen, birgt die Gefahr, dass der Ablauf dadurch unübersichtlich wird. Es ist wichtig bei der Verwendung von `Goto` auf die Verständlichkeit zu achten. Dabei helfen aussagekräftige Marken (sprechen aus, für was sie stehen).

3.7.2 Verzweigung mittels IF

Die `IF` Anweisung ermöglicht es den weiteren Ablauf auf der Basis einer Bedingung festzulegen. Entweder man geht den einen oder anderen Weg. (Verzweigung).

Das Hinzufügen eines `NOT` ermöglicht das Umkehren der Bedingung. Nicht Existenz, Ungleichheit oder dass nicht der Rückgabewert erhalten wurde.

Es werden drei verschiedene Möglichkeiten von Bedingungen unterschieden.

- Überprüfung der Existenz einer Datei

```
IF [NOT] EXIST Dateiname Befehl
```

```
IF_DateiExistiert.bat
1  @ECHO OFF
2  REM Ueberpruefe Existenz einer Datei
3  IF NOT EXIST beispiel.txt ECHO Datei existiert nicht.
```

```
C:\WORK\M122>IF_DateiExistiert.bat
Datei existiert nicht.
C:\WORK\M122>
```

- Vergleich zweier Zeichenketten

IF [NOT] "Zeichenkette" == "Zeichenkette2" Dateiname Befehl

```
IF_Zeichenkette.bat
1 @ECHO OFF
2 REM Vergleiche den dem Script übergebenen Parameter
3 REM mit dem String (Text) Hallo
4 IF "%1"=="Hallo" ECHO Parameter war Hallo.
```

(%1 steht für den ersten Parameter des Aufrufs der Stapeldatei.)

```
C:\WORK\M122>IF_Zeichenkette.bat Hallo
Parameter war Hallo.
C:\WORK\M122>
```

- Das Prüfen eines Rückgabewertes eines aufgerufenen Programmes

IF [NOT] ERRORLEVEL Wert GOTO

```
IF_Errorlevel.bat
1 @ECHO OFF
2 rem Vergleiche Rueckgabewerte
3 find "%1" "%2" > NUL
4 IF ERRORLEVEL 1 GOTO NICHTS
5 IF ERRORLEVEL 0 GOTO GEFUNDEN
6 GOTO END
7
8 :NICHTS
9 ECHO Nicht gefunden :- (
10 GOTO END
11
12 :GEFUNDEN
13 ECHO Gefunden :- )
14 GOTO END
15
16 :END
```

(%1, %2 stehen für den ersten und zweiten Parameter des Aufrufs der Stapeldatei.)

```
C:\WORK\M122>IF_Errorlevel.bat Pause 02_PauseRem.bat
Gefunden :- )
```

ACHTUNG: IF ERRORLEVEL <zahl> prüft auf >= zahl !!!
Damit muss zuerst der grössere Wert geprüft werden!!

Anwendung von IF und ELSE

```
IF_DateiExistiert_1.bat
1 @ECHO OFF
2 REM Ueberpruefe Existenz einer Datei
3 IF NOT EXIST beispiel.txt (
4 ECHO Datei existiert nicht
5 ) ELSE (
6 ECHO Datei existiert
7 )
```

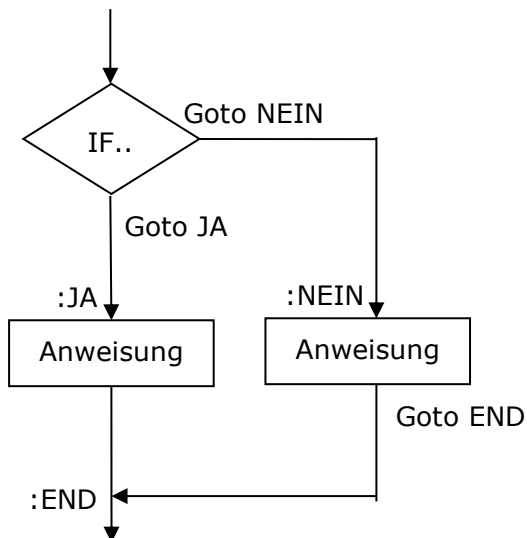
```
C:\WORK\M122>dir beispiel.txt
Volume in drive C is system
Volume Serial Number is EE3A-CAD9

Directory of C:\WORK\M122

File Not Found

C:\WORK\M122>IF_DateiExistiert_1.bat
Datei existiert nicht
```

Oft verwendet man an Stelle von ELSE mehrere GOTO und Label.



```
IF_DateiExistiert_2.bat
1 @ECHO OFF
2 rem Ueberpruefe Existenz einer Datei
3 IF NOT EXIST beispiel.txt GOTO NOTEXISTS
4 GOTO EXISTS
5
6 :NOTEXISTS
7 ECHO Datei existiert nicht
8 GOTO END
9
10 :EXISTS
11 ECHO Datei existiert
12
13 :END
```

```
C:\WORK\M122>dir beispiel.txt
Volume in drive C is system
Volume Serial Number is EE3A-CAD9

Directory of C:\WORK\M122

19.05.2013  13:29                3 beispiel.txt
               1 File(s)                3 bytes
               0 Dir(s)  66'486'636'544 bytes free

C:\WORK\M122>IF_DateiExistiert_2.bat beispiel.txt
Datei existiert
```

3.7.3 Wiederholung mittels FOR

Die Basis der FOR Anweisung ist eine Liste mit einer bestimmten Anzahl von Elementen. Die FOR-Anweisung arbeitet diese Liste ab. Es resultiert eine Schleife, die so oft durchlaufen wird, wie die Liste Elemente vorweist.

```
FOR_Liste.bat
1 @ECHO OFF
2 REM alle Element der Liste ausgeben
3 FOR %%f IN (A B C D E) DO (
4     ECHO Element: %%f
5 )
```

```
C:\WORK\M122>FOR_Liste.bat
Element: A
Element: B
Element: C
Element: D
Element: E
```

%%f ist die Schleifen-Variable für das aktuell in der Schleife verwendete Element aus der Liste. In jedem Durchlauf ändert das Element.

Als Zähl-Variablen müssen immer **einbuchstabige** Namen gewählt werden. Diese werden mit innerhalb von Batch-Dateien mit zwei % Zeichen versehen sind. Ausserhalb von Batch-Dateien genügt ein %.

Mit Hilfe von FOR kann man auch eine Liste von Dateien bearbeiten

zBsp alle Batch Dateien: *.bat

Der * ist ein Wildcard, er steht für eine beliebige Folge von Zeichen. Mit Hilfe des Wildcard und der Dateierdung *.bat wird die Liste aller Batch Dateien im aktuellen Verzeichnis erstellt. Jede Batch-Datei ist also ein Element in der Liste.

Die Beispiel Batch-Datei kopiert jede Batch-Datei in eine Datei gleichen Namens jedoch mit der Endung .bak (Backup).

```
FOR_Verzeichnis.bat
1 @ECHO OFF
2 REM Erstelle von allen Batch Dateien eine Kopie.
3 REM Die erstellten Dateien haben die Endung bak.
4 FOR %%a IN (*.bat) DO COPY %%a *.bak
```

```
C:\WORK\M122>FOR_Verzeichnis.bat
1 file(s) copied.
1 file(s) copied.
1 file(s) copied.
```

FOR kennt verschiedene Optionen und Parameter, siehe dazu die Hilfe zu FOR

Ein Beispiel dazu, wenn man in einem bestimmten Verzeichnis die Aktion ausführen will, benützt man die Option /R.

```
FOR_Verzeichnis_2.bat
1 @ECHO OFF
2 REM Erstelle von allen Batch Dateien eine Kopie.
3 REM Die erstellten Dateien haben die Endung bak.
4 FOR /R C:\WORK\M122\Test %%a IN (*.bat) DO COPY %%a *.bak
```

3.7.4 Auswahl mittels CHOICE

Um auch so etwas wie ein Menü realisieren zu können, steht der Befehl *CHOICE* zur Verfügung. *CHOICE* erwartet als Parameter einen Satz, der auf dem Bildschirm ausgegeben wird und eine Aufzählung von Buchstaben, die als Antwort gültig sind.

```

1 @ECHO OFF
2 ECHO Make your choice (Treffen Sie ihre Wahl).
3 CHOICE /C YNC /M "Press Y for Yes, N for No or C for Cancel."
4 IF ERRORLEVEL 3 GOTO CANCEL
5 IF ERRORLEVEL 2 GOTO NO
6 IF ERRORLEVEL 1 GOTO YES
7 GOTO END
8
9 :YES
10 ECHO Yes, I can.
11 GOTO END
12
13 :NO
14 ECHO No, I can not.
15 GOTO END
16
17 :CANCEL
18 ECHO Cancel...
19
20 :END
  
```

```

C:\WORK\M122>Make_your_CHOICE.bat
Make your choice (Treffen Sie ihre Wahl).
Press Y for Yes, N for No or C for Cancel. [Y,N,C]?Y
Yes, I can.
  
```

Wird die Aufzählung der Tasten weglassen, so wird standardmässig die Auswahl J und N angenommen (bei englischem DOS: Y and N).

Es ist auch möglich, ein *Timeout* einzustellen und zu definieren, welche Taste nach einer bestimmten Zeit als gedrückt gelten soll. Dazu ist der Parameter */T c,nn* anzugeben, wobei hier *c* für die angenommene Taste steht und *nn* für die Anzahl der Sekunden, die auf eine Eingabe gewartet werden soll.

Durch Angabe des Parameters */S* wird zwischen Gross- und Kleinschreibung unterschieden. Der Parameter */N+* unterdrückt die Abfrage, stellt aber den Text dar.

3.8 Kommandozeilen Parameter

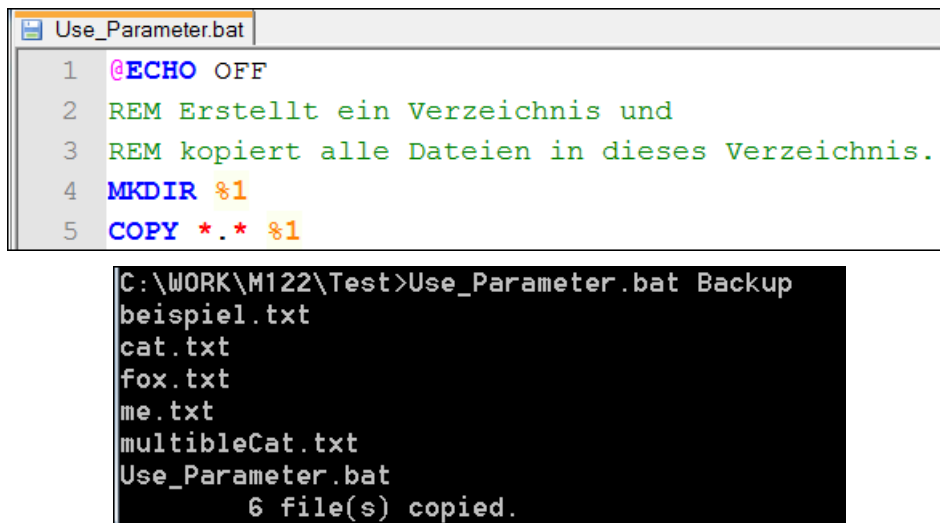
Batch-Dateien kann wie anderen Programmen auch Parameter beim ihrem Aufruf mitgegeben werden. Diese Parameter können von der Batch-Datei bearbeitet werden. Die Parameter werden innerhalb der Datei mit den Namen %1 bis %9 bezeichnet.

%0 steht für den Namen der Batch-Datei

%1 steht für den ersten Parameter

%2 steht für den zweiten Parameter usw.

Im Beispiel wird ein Verzeichnis mit dem als Parameter übergebenen Namen erstellt. Danach werden alle Dateien aus dem aktuellen Verzeichnis in das neue Verzeichnis kopiert.



```
Use_Parameter.bat
1 @ECHO OFF
2 REM Erstellt ein Verzeichnis und
3 REM kopiert alle Dateien in dieses Verzeichnis.
4 MKDIR %1
5 COPY *.* %1

C:\WORK\M122\Test>Use_Parameter.bat Backup
beispiel.txt
cat.txt
fox.txt
me.txt
multibleCat.txt
Use_Parameter.bat
        6 file(s) copied.
```

3.8.1 Mehr als 9 Parameter

Mit einem Trick und dem Befehl *SHIFT* können auch mehr als neun Parameter benutzt werden. Die können aber nicht einzeln mit Nummern angesprochen werden. *SHIFT* verschiebt einfach alle Parameter um eins nach links. Das heisst, der erste Parameter fällt weg, der zweite wird zum ersten, der dritte zum zweiten usw. Damit ist auch die Abarbeitung aller Parameter möglich, auch wenn Sie gar nicht wissen, wie viele eigentlich benutzt wurden.

3.9 Variablen

Variablen Zwischenspeicher von Werten. Variablen haben einen Namen, dieser ist idealerweise sprechend, damit man die Bedeutung der Variablen erkennen kann.

Windows kennt die Möglichkeit, Umgebungsvariablen zu definieren. Zum einen gibt es die Umgebungsvariablen, die im ganzen System Geltung haben. Zum anderen können in einem Kommandointerpreter neue Umgebungsvariablen definiert werden. Deren Gültigkeit beschränkt sich dann auf den Kontext des Kommandointerpreters und auf die aus ihm gestarteten Programme.

Eines der bekanntesten Beispiele ist die PATH Variable, in der der Suchpfad angegeben wird, der nach ausführbaren Dateien durchsucht werden soll.

Welche Umgebungsvariablen bereits definiert sind, kann mittels *SET* ausgegeben werden.

Im Beispiel sieht man nur die ersten paar definierten Umgebungsvariablen, die Liste ist nicht vollständig.

```
C:\WORK\M122>set
ALLUSERSPROFILE=C:\ProgramData
APPDATA=C:\Users\peter.rutschmann\AppData\Roaming
CommonProgramFiles=C:\Program Files\Common Files
CommonProgramFiles(x86)=C:\Program Files (x86)\Common Files
CommonProgramW6432=C:\Program Files\Common Files
COMPUTERNAME=N001
ComSpec=C:\Windows\system32\cmd.exe
FP_NO_HOST_CHECK=NO
HOMEDRIVE=C:
```

3.9.1 Setzen und Löschen von Umgebungsvariablen

Umgebungsvariablen wird mit Hilfe von *SET* ein Wert zugewiesen.
Sie dürfen auch Leerzeichen enthalten.

```
SET VARIABLE=WERT
```

Ohne Wert nach dem Gleichheitszeichen, wird die Umgebungsvariable gelöscht.

```
SET VARIABLE=
```

```
C:\WORK\M122>set Name
Environment variable Name not defined

C:\WORK\M122>set Name=Rutschmann

C:\WORK\M122>set Name
Name=Rutschmann

C:\WORK\M122>set Name=

C:\WORK\M122>set Name
Environment variable Name not defined
```

Der alte Wert einer Umgebungsvariablen kann mit *SET* mit einem neuen Wert überschrieben werden.

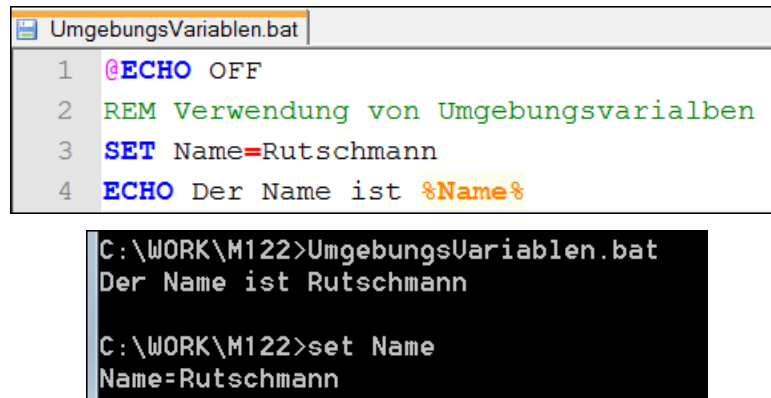
Neu gesetzte Umgebungsvariablen sind nur im Kommandointerpreter und in den aus ihm gestarteten Programmen gültig.

Systemweite Umgebungsvariablen müssen in den Einstellungen von Windows verändert werden. Nach einer Änderung der systemweit geltenden Umgebungsvariablen, ist es

empfehlenswert, sich neu bei Windows anzumelden, um sicher zu sein, dass die veränderten Umgebungsvariablen in den laufenden Programmen berücksichtigt werden.

3.9.2 Verwendung von Umgebungsvariablen

Will man eine Umgebungsvariable in einer Batch-Datei verwenden, muss der Variablenname innerhalb von Prozentzeichen (%) aufgeführt werden:



```

UmgebungsVariablen.bat
1 @ECHO OFF
2 REM Verwendung von Umgebungsvariablen
3 SET Name=Rutschmann
4 ECHO Der Name ist %Name%

C:\WORK\M122>UmgebungsVariablen.bat
Der Name ist Rutschmann

C:\WORK\M122>set Name
Name=Rutschmann
    
```

3.10 Starten und Aufrufen

3.10.1 Einen neuen Kommandointerpreter starten

Jederzeit kann aus einem Kommandointerpreter ein neuer Kommandointerpreter gestartet werden. Dieser Kommandointerpreter **startet im gleichen Fenster**.

CMD

Oder

Command

Dabei erbt der neu gestartete Kommandointerpreter alle Umgebungsvariablen des übergeordneten Kommandointerpreters.

Es können nun unabhängig Änderungen an den Variablen und ihren Einstellungen vorgenommen werden. Das heisst, werden nach dem Start des zweiten Kommandointerpreters Umgebungsvariablen verändert, so haben diese Änderungen für den anderen Kommandointerpreter keine Gültigkeit.

Der aktuelle Kommandointerpreter wird mit **EXIT** verlassen. Dabei gehen alle in ihm neu definierten Variablen verloren!

3.10.2 Ein neues Kommandointerpreter Fenster starten

Mittels **START** kann **ein neues** Kommandointerpreter **Fenster** gestartet werden. Dieses Fenster ist nun ein vom alten Fenster unabhängiger Prozess und arbeitet parallel zum anderen Fenster.

Der Befehl **START** verfügt über eine Vielzahl von Einstellmöglichkeiten. Siehe *start /?*

Das Fenster kann mittels **EXIT** geschlossen werden.

Mit START kann auch ein Programm parallel zum laufenden Kommandointerpreter gestartet werden. Ohne START läuft das Programm im aktuellen Kommandointerpreter und blockiert diesen.

Starten eines Explorers:

```
START explorer
```

Starten eines Programmes aus einem bestimmten Verzeichnis:

```
START /D "C:\Program Files (x86)\Notepad++" notepad++.exe
```

3.10.3 Eine andere Batch-Datei starten

Mit CALL können Sie aus einer Batch-Datei heraus andere Batch-Dateien aufrufen.

Ohne CALL wird die zuerst gestartete Batch-Datei abgebrochen.

```
CallAndStart.bat
1 @ECHO OFF
2 ECHO Notepad in einem neuen Fenster starten
3 start notepad
4 ECHO Batch-Datei in einem neuen Fenster starten
5 start 02_PauseRem.bat
6 ECHO Batch-Datei starten und nach der Bearbeitung
7 ECHO zurueck kehren
8 call 02_PauseRem.bat
9 ECHO Notepad starten, Batch Datei bleibt blockiert
10 ECHO bis Notepad geschlossen wird.
11 notepad
12 ECHO Batch-Datei starten. Aktuelle Batchdatei wird
13 ECHO nicht mehr fortgesetzt
14 02_PauseRem.bat
15 ECHO deshalb wird dieses Echo nie ausgegeben.
```

```
C:\WORK\M122>CallAndStart.bat
Notepad in einem neuen Fenster starten
Batch-Datei in einem neuen Fenster starten
Batch-Datei starten und nach der Bearbeitung
zurueck kehren
Moment mal
Press any key to continue . . .
Ich warte auf Ihre naechste Eingabe
Notepad starten, Batch Datei bleibt blockiert
bis Notepad geschlossen wird.
Batch-Datei starten. Aktuelle Batchdatei wird
nicht mehr fortgesetzt
Moment mal
Press any key to continue . . .
Ich warte auf Ihre naechste Eingabe
C:\WORK\M122>
```

3.11 Ausgaben umleiten

Viele Programme, Build-Ins und Utilities machen Ausgaben im Fenster des Kommandointerpreters (Konsole). Andere Programme warten auf die eine Eingabe über die Tastatur.

Standardmässig sind drei Wege (Kanäle) für die Ein- und Ausgabe definiert:

Kanal	Nummer	Beschreibung
STDIN	0	Standard Eingabe: Eingabe über die Tastatur als Eingabe für ein Programm. Zum Beispiel verlangt der Befehl <i>DEL</i> vor dem Löschen einer Datei eine Ja/Nein Bestätigung über eine Eingabe per Tastatur.
STDOUT	1	Standard Ausgabe: Die Ausgabe eines Programms wird im Fenster des Kommandointerpreters (Konsole) ausgegeben. So listet der Befehl <i>DIR</i> alle Verzeichnisse und Dateien in einem Verzeichnis auf.
STDERR	2	Ausgabe Kanal für Fehler: Entsteht bei der Verarbeitung eines Befehls ein Fehler, so geben die Programme Fehlermeldungen in das Fenster des Kommandointerpreters aus. So zum Beispiel, wenn beim Aufruf von <i>DIR</i> eine unbekannte Option /Z mitgegeben wurde.

Oft will man diese Ausgaben Umleiten, zBps um eine Liste aller Dateien in einem Verzeichnis zu erhalten, leitet man die Ausgabe des Befehls *DIR* in eine Datei um.

```

Umleitung.bat
1 @ECHO OFF
2 REM Leite Die Ausgabe von DIR in eine Datei um.
3 DIR > temp.txt

```

Es sind folgende Umleitungsoperator definiert:

Umleitungsoperator	Bedeutung
>	Die Ausgabe wird in eine neue Datei geschrieben.
>>	Die Ausgabe wird an eine bestehende Datei angehängt. Ist die Datei nicht vorhanden, so wird diese erzeugt.
<	Liest die Eingabe von einer Datei anstelle von der Tastatur.
>&	Schreibt die Ausgabe von einem Kanal in einen anderen umleiten. (Siehe Beschreibung weiter unten)
<&	Liest die Daten von einem Kanal und gibt dies an einen anderen weiter. (Siehe Beschreibung weiter unten)
	Die Ausgabe des ersten Befehls dient als Eingabe des nächsten Befehls. Man gibt die Ausgabe über eine <i>Pipe</i> an den nächsten Befehl weiter.

Beispiel:

```
DIR :c\ >>DIRECTORY.LOG 2>&1
```

Im Beispiel wird der Inhalt des aktuellen Verzeichnisses in der Datei DIRECTORY.LOG festgehalten.

→ Umleitung des STDOUT in die Datei DIRECTORY.LOG mittels >>.

Da sich im Aufruf aber ein Fehler befindet, entsteht eine Fehlermeldung. Diese Fehlermeldung des Fehlers wird, dank der zweiten Umleitung 2>%1, anstelle auf dem Bildschirm in den STDOUT und damit in die Datei DIRECTORY.LOG geschrieben.

Beispiel:

```
DIR | SORT
```

Hier wird die Ausgabe von DIR direkt als Eingabe von SORT verwendet. Es wird als der Inhalt des aktuellen Verzeichnis (Sub-Verzeichnisse und Dateien) sortiert auf dem Bildschirm ausgegeben.

Über solche Verkettung entstehen mächtige Arbeitsketten. Was mitunter auch eine Kunst der Batch-Programmierung ist.

3.12 Weitere Operatoren

Mittels **&** lassen sich mehrere Batch-Befehle in einer Zeile hintereinander ausführen.

```
Echo Hallo! & echo und Adieu!
```

Ausgabe:

```
Hallo!  
und Adieu!
```

Mittels **&&** wird der zweite Befehl nur ausgeführt, wenn der erste Befehl erfolgreich war.

```
Copy myfile.txt myfile.bak && echo Backup erfolgreich erstellt.
```

Ausgabe sofern kopieren gelungen ist:

```
Backup erfolgreich erstellt
```

Mittels **||** wird der zweite Befehl nur ausgeführt, wenn der erste Befehl fehlschlug.

```
Copy myfile.txt myfile.bak || echo Backup fehlgeschlagen.
```

Ausgabe sofern kopieren nicht gelungen ist:

```
Backup fehlgeschlagen
```

4. Übersicht über die Befehle von cmd

Gibt man im Kommandointerpreter *CMD* Help ein, so erscheint die folgende Liste. Um mehr Informationen zu einem einzelnen Befehl zu erhalten, gibt man *HELP NameDesBefehls* ein.

ASSOC	Displays or modifies file extension associations.
ATTRIB	Displays or changes file attributes.
BREAK	Sets or clears extended CTRL+C checking.
BCDEDIT	Sets properties in boot database to control boot loading.
CACLS	Displays or modifies access control lists (ACLs) of files.
CALL	Calls one batch program from another.
CD	Displays the name of or changes the current directory.
CHCP	Displays or sets the active code page number.
CHDIR	Displays the name of or changes the current directory.
CHKDSK	Checks a disk and displays a status report.
CHKNTFS	Displays or modifies the checking of disk at boot time.
CLS	Clears the screen.
CMD	Starts a new instance of the Windows command interpreter.
COLOR	Sets the default console foreground and background colors.
COMP	Compares the contents of two files or sets of files.
COMPACT	Displays or alters the compression of files on NTFS partitions.
CONVERT	Converts FAT volumes to NTFS. You cannot convert the current drive.
COPY	Copies one or more files to another location.
DATE	Displays or sets the date.
DEL	Deletes one or more files.
DIR	Displays a list of files and subdirectories in a directory.
DISKCOMP	Compares the contents of two floppy disks.
DISKCOPY	Copies the contents of one floppy disk to another.
DISKPART	Displays or configures Disk Partition properties.
DOSKEY	Edits command lines, recalls Windows commands, and creates macros.
DRIVERQUERY	Displays current device driver status and properties.
ECHO	Displays messages, or turns command echoing on or off.
ENDLOCAL	Ends localization of environment changes in a batch file.
ERASE	Deletes one or more files.
EXIT	Quits the CMD.EXE program (command interpreter).

FC	Compares two files or sets of files, and displays the differences between them.
FIND	Searches for a text string in a file or files.
FINDSTR	Searches for strings in files.
FOR	Runs a specified command for each file in a set of files.
FORMAT	Formats a disk for use with Windows.
FSUTIL	Displays or configures the file system properties.
FTYPE	Displays or modifies file types used in file extension associations.
GOTO	Directs the Windows command interpreter to a labeled line in a batch program.
GPRESULT	Displays Group Policy information for machine or user.
GRAFTABL	Enables Windows to display an extended character set in graphics mode.
HELP	Provides Help information for Windows commands.
ICACLS	Display, modify, backup, or restore ACLs for files a directories.
IF	Performs conditional processing in batch programs.
LABEL	Creates, changes, or deletes the volume label of a disk.
MD	Creates a directory.
MKDIR	Creates a directory.
MKLINK	Creates Symbolic Links and Hard Links
MODE	Configures a system device.
MORE	Displays output one screen at a time.
MOVE	Moves one or more files from one directory to another directory.
OPENFILES	Displays files opened by remote users for a file share.
PATH	Displays or sets a search path for executable files.
PAUSE	Suspends processing of a batch file and displays a message.
POPD	Restores the previous value of the current directory saved by PUSHG.
PRINT	Prints a text file.
PROMPT	Changes the Windows command prompt.
PUSHD	Saves the current directory then changes it.
RD	Removes a directory.
RECOVER	Recovers readable information from a bad or defective disk.
REM	Records comments (remarks) in batch files or CONFIG.SYS.
REN	Renames a file or files.
RENAME	Renames a file or files.
REPLACE	Replaces files.
RMDIR	Removes a directory.

ROBOCOPY	Advanced utility to copy files and directory trees
SET	Displays, sets, or removes Windows environment variables.
SETLOCAL	Begins localization of environment changes in a batch file.
SC	Displays or configures services (background processes).
SCHTASKS	Schedules commands and programs to run on a computer.
SHIFT	Shifts the position of replaceable parameters in batch files.
SHUTDOWN	Allows proper local or remote shutdown of machine.
SORT	Sorts input.
START	Starts a separate window to run a specified program or command.
SUBST	Associates a path with a drive letter.
SYSTEMINFO	Displays machine specific properties and configuration.
TASKLIST	Displays all currently running tasks including services.
TASKKILL	Kill or stop a running process or application.
TIME	Displays or sets the system time.
TITLE	Sets the window title for a CMD.EXE session.
TREE	Graphically displays the directory structure of a drive or path.
TYPE	Displays the contents of a text file.
VER	Displays the Windows version.
VERIFY	Tells Windows whether to verify that your files are written correctly to a disk.
VOL	Displays a disk volume label and serial number.
XCOPY	Copies files and directory trees.
WMIC	Displays WMI information inside interactive command shell.