Modul 122 - Teil 1

(Workshop Grundlagen zum Thema DOS - Batch)

Inhaltsverzeichnis:

1 Einführung 2
2 DOS Batch - Programmierung 2
2.1 Grundlagen 2
2.2 Aufbau 3
2.2.1 Abarbeiten eines Befehls-Stapels: 3
2.2.2 Textausgabe mit ECHO: 3
2.2.3 Dateiumleitung 4
2.2.4 Anhalten des Ablaufs mit PAUSE: 4
2.2.5 Kommentare: 5
2.2.6 Variablen 5
2.2.7 Neuer Kommandointerpreter 5
2.2.8 Kontrollstrukturen 5
2.2.9 CHOICE 7
2.2.10 FOR 7
2.2.11 Kommandozeilenparameter 7
2.2.12 Zusammenfassung der Syntax für alle Batch-Befehle 8
2.3 Internetlinks 10
2.4 DOS – Aufgaben 11
2.5 DOS - Befehle (WINDOWS 2000) 19

Dieser Block ist als Workshop konzipiert. Lesen Sie das Skript aufmerksam durch und versuchen Sie danach selbstständig die Fragen und Aufgaben zu lösen.

Lernziele:

- DOS – Grundkenntnisse erwerben
- Batch Programmierung einsetzen können

1 Einführung

Die ersten Personal Computer hatten nur einfache Bedienungsprogramme, sogenannte Kommandozeilen-Interpreter. Diese hatten die Aufgabe Eingaben des Benutzers in Aktionen umzusetzen und andere Programme zu starten. Diese Kommandozeilen-Interpreter waren 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. In diesem Workshop soll jedoch der Kommandozeilen-Interpreter von Microsoft vorgestellt werden. Dieser hat den Namen *COMMAND.COM (DOS, Win95/98/ME)* respektive *CMD.EXE* (NT-Systeme).

Von Anfang an war auch das Bedürfnis da, auf einen Befehl nicht nur eine, sondern eine ganze Reihe vom Benutzer definierte Aktionen auszuführen. Die Aufgaben resp. DOS-Befehle werden dabei in einfache Text-Dateien geschrieben, welche anstelle des Benutzers dann die Eingaben für den Kommandointerperter liefern. Diese Programme werden BATCH-*Programme* genannt und haben unter DOS die Endung *.BAT*. Diese Art der Datenverarbeitung war man sich auch schon von frühen Grossrechnern gewöhnt, denen man gelochte Kartenstapel zur Steuerung fütterte, woraus sich dann auch 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 WindowsNT-Systeme stellen beide Interpreter zu Verfügung, während die älteren Windowssysteme basierend auf Windows 98/ME nur den COMMAND.COM Interpreter besitzen.

Der Kommandointerpreter CMD.EXE ist nur auf NT Betriebsystemen vorhanden!

2 DOS Batch - Programmierung

2.1 Grundlagen

Viele Kommandozeilen Programme verfügen über eine eingebaute Hilfe, welche zeigt wie das Programm aufgerufen werden soll. Dazu wird das jeweilige Programm mit dem Parameter */?* Aufgerufen. Genau so ist es mit dem Kommandozeileninterpreter.

```
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).
```

Die NT-Betriebssysteme 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 Batch-Programme bevor sie ausgeliefert werden auf den jeweiligen Betriebssystemen getestet werden.
Bsp. Unter Windows ME kann der Kommandointerperter Batch-Dateien schrittweise ausführen. Dazu wird ein neuer Kommandointerpreter mit dem Parameter */Y* und der Angabe des auszuführenden Batches aufgerufen (Dies funktioniert unter W2000/XP nicht!):

```
COMMAND /Y /C Test.bat
```

Viele der grundlegenden Befehle wie *dir*, *cd*, etc. sind direkt im Kommandointerperter direkt implementiert. Neuere Erweiterungen werden jedoch als externe Programme realisiert und müssen im Such-Pfad sein, damit sie gefunden werden (Bsp. CHOICE.EXE siehe weiter unten). Auch hier kann es zu Abweichungen kommen.

Zum Öffnen des Kommandozeilen-Interpreters 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.

2.2 Aufbau

2.2.1 Abarbeiten eines Befehls-Stapels:

BATCH-Dateien sind reine Textdateien (ASCII) und können mit jedem beliebigen Editor (z.B. EDIT) erstellt werden. Im einfachsten Fall enthalten sie, wie schon gesagt, nur mindestens einen DOS-Befehl:

```
CLS
DIR /p
```

Damit ist es also leicht möglich, komplizierte Befehle mit vielen Parametern einfach zugänglich zu machen, ohne sich die Parameter einzeln merken zu müssen. Um z.B. die Einstellung eines seriellen Druckers mit *MODE* zu vereinfachen könnten Sie in eine Datei *DRUCKER.BAT* schreiben:

```
MODE COM1 baud=9600 parity=N data=8 stop=1
MODE LPT1=COM1
```

Somit würde es genügen den Befehl *DRUCKER*, statt jedes Mal die beiden obigen Zeilen einzugeben.

2.2.2 Textausgabe mit ECHO:

Wenn eine Batch-Datei abgearbeitet wird, so werden auch 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*. Dabei schreiben Sie:

```
ECHO OFF
```

um das Echo (die Darstellung des Befehls auf dem Bildschirm) auszuschalten und

```
ECHO ON
```

um es wieder zu aktivieren. Üblicherweise wird das Echo in Batch-Dateien gleich zu Anfang abgeschaltet. Damit aber der Befehl ECHO OFF nicht selbst trotzdem dargestellt wird, wird ihm ein Klammeraffe (@) vorgestellt, also:

```
@ECHO OFF
```

Der Befehl ECHO wird aber auch 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. (Ausser ON und OFF) Dazu ist es nicht nur möglich, sondern auch geradezu notwendig, das ECHO vorher abzuschalten weil sonst der Befehl ECHO TEXT und in der nächsten Zeile erst der Text selbst auf dem Schirm erscheinen würde:

```
@ECHO OFF
ECHO Das ist eine erste Batchdatei.
...
```

Ohne jeden Parameter gibt *ECHO* den jeweiligen Zustand (ON oder OFF) aus, nicht etwa eine Leerzeile. Um eine Leerzeile auszugeben müssen Sie den *ECHO*-Befehl mit direkt folgendem Punkt angeben also:

```
ECHO.
```

2.2.3 Dateiumleitung

Die Ausgabe lässt sich leicht in eine Datei anstelle des Bildschirmes umleiten. Dies ist vorallem bei Batch-Jobs interessant die unbewacht ablaufen und dabei sich selbst protokollieren sollen. Die Ausgabe wird wie im folgenden Beispiel umgeleitet:

```
DIR >DIRECTORY.LOG
```

Dabei wird der Inhalt des Verzeichnisses anstelle auf dem Bildschirm in die Datei DIRECTORY.LOG geschrieben. Dabei Interessant ist, dass es eine spezielle Datei `nul` gibt. Wird die Ausgabe in diese Datei umgeleitet, so werden alle Ausgaben weggeworfen.

Umleitungsopertoren	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 der Tastatur.
>&	Schreibt die Ausgabe von einem Kanal in einen anderen. (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.

Folgende Daten-Kanäle sind definiert

Kanal	Nummer	Beschreibung
STDIN	0	Tastatur Eingang
STDOUT	1	Standard Ausgabe
STDERR	2	Ausgabe Kanal für Fehler

Damit lassen sich nun zum Beispiel die Fehler an den selben Ort ausgeben wie alle anderen Ausgaben, ansonsten wird der Fehler auf dem Bildschirm ausgegeben. Im folgenden Beispiel ist absichtlich ein Fehler enthalten. Spielen sie mal mit der Umleitung.

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

Ein weiteres Beispiel soll zeigen, wie die Ausgabe der einen Datei, als Eingabe der anderen Datei dienen soll.

```
DIR | SORT
```

2.2.4 Anhalten des Ablaufs mit PAUSE:

Oft ist es nötig, die Abarbeitung des Programms an einer bestimmten Stelle anzuhalten um z.B. auf einen Diskettenwechsel zu warten. Das geschieht mit dem Befehl *PAUSE*. Dabei wird eine Meldung ausgegeben, die besagt, dass eine Taste gedrückt werden soll, wenn es weiter gehen soll. Eine Batch Datei, die den Inhalt dreier Disketten in ein Verzeichnis kopiert könnte also z.B. so aussehen:

```
  @ECHO OFF
ECHO Bitte die erste Diskette einlegen...
PAUSE
COPY a:\*.* c:\VERZ1

ECHO Bitte die zweite Diskette einlegen...
PAUSE
COPY a:\*.* c:\VERZ1

ECHO Bitte die dritte Diskette einlegen...
PAUSE
COPY a:\*.* c:\VERZ1

ECHO Fertig...
```

2.2.5 Kommentare:

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.

2.2.6 Variablen

MS-DOS kennt die Möglichkeit, Umgebungsvariablen zu definieren, eins der bekanntesten Beispiele ist die PATH Variable, in der der Suchpfad angegeben wird, der nach ausführbaren Dateien durchsucht werden soll. Es gibt mehrere solcher Variablen etwa TEMP, COMSPEC. Auf diese Variablen kann in einer Batch-Datei zugegriffen werden. Dazu muss der Variablennamen innerhalb von Prozentzeichen (%) aufgeführt werden:

```
@ECHO OFF
ECHO Die Pfad Variable PATH enthält die Einträge:
ECHO %PATH%
ECHO Als Kommandointerpreter wird %COMSPEC% benutzt.
```

Solche Variablen können beliebig gesetzt werden, mit dem Befehl SET VARIABLE=WERT werden sie erzeugt, mit einem einfachen SET VARIABLE= werden sie wieder gelöscht. Sie dürfen auch Leerzeichen enthalten:

```
@ECHO OFF
SET VAR1=Hallo
SET VAR2=Hans Hugo Häberlein
ECHO %VAR1% %VAR2%
SET VAR1=
SET VAR2=
```

Vorsicht: Die bereits bestehenden Umgebungsvariablen wie PATH oder TEMP sollten natürlich nicht gelöscht werden, wenn nicht erwünscht ist, dass sie nach Ablauf der Batch-Datei weiter gelöscht bleiben.

Anmerkung: Müssen sie eine der Standard Variablen temporär ändern, so starten sie einfach einen neuen Kommandointerpreter. Jedes Programm bekommt die Parameter seines Aufrufers als Kopie mit und kann diese verändern, ohne das übergeordnete Programm zu stören.

2.2.7 Neuer Kommandointerpreter

Jederzeit kann ein neuer Kommandointerpreter aus einem anderen gestartet werden. Dabei erbt der neue Interpreter alle Einstellungen des übergeordneten Interpreters. Es können nun unabhängig Änderungen an den Variablen und ihren Einstellungen vorgenommen werden. Der aktuelle Kommandointerpreter wird mit *EXIT* verlassen. Dabei gehen alle seine Variablen verloren!

2.2.8 Kontrollstrukturen

Kontrollstrukturen sind Möglichkeiten, den linearen Ablauf einer Batch-Datei zu verändern. Dazu stehen hier leider nur wenige zur Verfügung, es reicht aber für die wichtigsten Aufgaben.

Goto

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 bezeichnet und mit einem Doppelpunkt und einem Markennamen gebildet. So lassen sich z.B. Endlosschleifen programmieren:

```
@ECHO OFF
:Anfang
ECHO in der Schleife
GOTO Anfang
```

Verzweigung (IF)

Mit der *IF* Anweisung können Sie drei verschiedene Bedingungen überprüfen, die Existenz einer Datei, die Gleichheit zweier Zeichenketten und den Rückgabewert des zuletzt gelaufenen DOS-Programms (ERRORLEVEL). Durch das Zufügen eines *NOT* können auch die jeweils gegenteiligen Bedingungen Überprüft werden. (Nichtexistenz, Ungleichheit oder die

Tatsache dass ein Programm NICHT mit einem bestimmten Rückgabewert beendet wurde) In der Regel werden IF-Anweisungen mit dem GOTO Befehl verbunden, sie können jedoch auch einen einzigen Befehl veranlassen.

- ***Existenz von Dateien überprüfen***

  Syntax: IF [NOT] EXIST *Dateiname Befehl*

  Ein Beispiel, das eine Datei nur dann kopiert, wenn sie existiert und sonst eine Fehlermeldung ausgibt wäre etwa:

```
@ECHO OFF
IF NOT EXIST test.dat GOTO fehler
COPY test.dat c:\verz1
GOTO ende
:fehler
ECHO Datei TEST.DAT nicht gefunden !
:ende
```

  Tipp: Um zu überprüfen, ob ein Verzeichnis existiert, können Sie einfach die Existenz
  der Datei *NULL* in diesem Verzeichnis abfragen. Existiert sie, so existiert auch das Verzeichnis.

- ***Zeichenkettenvergleich***

  Syntax: IF [NOT] "*Zeichenkette1*"=="*Zeichenkette2*" *Befehl*

  Hier können zwei Zeichenketten miteinander verglichen werden, beide sollten immer in doppelten Anführungszeichen stehen. Als Zeichenketten können sowohl eingetippte Wörter, Umgebungsvariablen (siehe unten) als auch Kommandozeilenparameter (siehe auch weiter unten) benutzt werden. Ein kurzes Beispiel, das überprüft ob ein Kommandozeilenparameter eingegeben wurde:

```
@ECHO OFF
IF "%1"=="" GOTO fehler
...
GOTO ende
:fehler
ECHO Keine Parameter eingegeben...
:ende
```

- ***ERRORLEVEL überprüfen***

  Syntax: IF [NOT] ERRORLEVEL *Wert Befehl*

  Jedes DOS Programm liefert beim Beenden einen Abschluss-Fehlercode. Falls das Programm fehlerfrei abgelaufen ist, so gibt es meist den Wert 0 zurück, falls es Fehler gab eine Fehlernummer. Diese Fehlernummer kann auch mittels *IF*-Anweisung abgefragt werden. Wichtig ist, dass hier immer mit der grössten Nummer angefangen wird, weil die *IF*-Anweisung komischerweise nicht auf Gleichheit, sondern auf grösser oder gleich prüft Ein Beispiel für diese Anwendung gibt es gleich im nächsten Abschnitt, CHOICE ist nämlich ein Programm, das sein Ergebnis über den ERRORCODE zurückgibt...

2.2.9 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. Also etwa:

```
@ECHO OFF
CHOICE /C:ABC Drücken sie die A, B oder C Taste...
IF ERRORLEVEL 3 ECHO C eingegeben.
IF ERRORLEVEL 2 ECHO B eingegeben.
IF ERRORLEVEL 1 ECHO A eingegeben.
...
```

In diesem Fall wird keine andere Taste als A, B oder C anerkannt. Sollten Sie die Aufzählung der Tasten weglassen, so wird standardmässig die Auswahl J und N angenommen (bei englischem DOS natürlich dann Y und 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.

2.2.10 FOR

Die *FOR*-Anweisung arbeitet eine Liste ab, es realisiert eine Schleife, die so oft durchlaufen wird, wie die Liste Elemente vorweist. Als Liste können sowohl Dateinamen mit Metazeichen (z.B. *.*) als auch Zeichenketten verwendet werden. Um sich etwa alle Textdateien im aktuellen Verzeichnis anzuschauen könnten Sie schreiben:

```
@ECHO OFF
FOR %%a IN (*.txt) DO type %%a
```

Als Variablen müssen Sie immer einbuchstabige Namen wählen, die mit zwei % Zeichen versehen sind. (Ausserhalb von Batch-Dateien genügt ein %)

2.2.11 Kommandozeilenparameter

Normale Parameter

Batch-Dateien können, wie alle anderen Programme auch Kommandozeilenparameter bearbeiten. Diese Parameter werden innerhalb der Datei mit den Namen %1 bis %9 bezeichnet. Um also etwa eine Batch-Datei zu erstellen, die den Inhalt einer Diskette in ein bestimmtes Verzeichnis kopiert, könnten Sie schreiben:

```
@ECHO OFF
ECHO Bitte die Diskette einlegen...
PAUSE
COPY A:\*.* %1
ECHO Fertig.
```

Würden Sie diese Datei *ACOPY.BAT* nennen und sie folgendermassen aufrufen:

```
ACOPY C:\VERZ1
```

so stünde in der Zeile mit dem Kopierbefehl jetzt statt dem %1 der angegebene Parameter C:\VERZ1 und der Befehl würde also lauten:

```
...
COPY A:\*.* C:\VERZ1
...
```

Es sind also genau neun Parameter mit dieser Methode ansprechbar, der Parameter %0 hat eine Sonderbedeutung, er enthält den Namen der Batch-Datei.

Mehr als 9 Parameter

Mit einem Trick und dem Befehl *SHIFT* können auch mehr als neun Parameter benutzt werden, die können dann 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 sind also auch die Abarbeitung aller Parameter möglich, auch wenn Sie gar nicht wissen, wie viele eigentlich benutzt wurden.

Hier ein Beispiel:
Das folgende Programm schreibt alle angegebenen Parameter auf den Bildschirm, keinen zuviel und aber auch keinen zuwenig...

```
@ECHO OFF
:anfang
IF "%1"=="" GOTO ende
ECHO %1
SHIFT
GOTO anfang
:ende
```

So ist es also auch möglich, Programme zu schreiben, die für alle Parameter das gleiche tun, ohne zu wissen, wieviele es denn sind. Es wird solange immer nur der erste Parameter ausgewertet, solange es einen ersten Parameter gibt. Nach jedem Durchgang wird mit *SHIFT* eine Verschiebung vorgenommen...

CALL

Mit *CALL* können Sie aus einer Batch-Datei heraus andere Batch-Dateien aufrufen. Machen Sie es einfach ohne Call, so funktioniert zwar der Aufruf der anderen Batch-Datei, nach deren Abarbeitung kehrt die Kontrolle nicht mehr zur aufrufenden Batch-Datei zurück.

2.2.12 Zusammenfassung der Syntax für alle Batch-Befehle

Die folgenden Kurzbeschreibungen sind der Online Hilfe von MS-DOS entnommen und können jederzeit durch die Eingabe *BEFEHL* /? abgerufen werden.

ECHO

Zeigt Meldungen an oder schaltet die Befehlsanzeige ein (ON) oder aus (OFF).
ECHO [ON | OFF]
ECHO [Meldung]
ECHO ohne Parameter zeigt die aktuelle Einstellung der Befehlsanzeige an.

PAUSE

Hält die Ausführung einer Stapelverarbeitungsdatei an und zeigt folgende Meldung an:

```
Eine beliebige Taste drücken, um fortzusetzen
```

IF

Verarbeitet Ausdrücke mit Bedingungen in einem Stapelverarbeitungsprogramm.
IF [NOT] ERRORLEVEL Nummer Befehl
IF [NOT] Zeichenfolge1==Zeichenfolge2 Befehl
IF [NOT] EXIST Dateiname Befehl

NOT	Der Befehl soll nur dann ausgeführt werden, wenn die Bedingung nicht erfüllt ist.
ERRORLEVEL Nummer	Diese Bedingung ist erfüllt, wenn das zuletzt ausgeführte Programm einen Code grösser oder gleich Nummer zurückliefert.
Befehl	Der auszuführende Befehl, falls die Bedingung erfüllt ist.
Zeichenfolge1==Zeichenfolge2	Diese Bedingung ist erfüllt, falls die Zeichenfolgen übereinstimmen.
EXIST Dateiname	Diese Bedingung ist erfüllt, falls die angegebene Datei existiert.

GOTO

Setzt die Ausführung eines Stapelverarbeitungsprogramms an einer Marke fort.
GOTO Marke
Marke Eine Zeichenfolge als Marke in einem Stapelverarbeitungsprogramm.
Marken stehen allein am Zeilenanfang mit einem vorangestellten Doppelpunkt.

FOR

Führt einen Befehl für jede einzelne Datei eines Satzes von Dateien aus.
FOR %%Variable IN (Satz) DO Befehl [Parameter]

%%Variable	Ein ersetzbarer Parameter (bestehend aus einem einzelnen Zeichen).
(Satz)	Ein Satz von mindestens einer Datei. Platzhalter sind zulässig.
Befehl	Befehl, der für jede Datei ausgeführt werden soll.
Parameter	Parameter und Optionen für den angegebenen Befehl.

CHOICE

Wartet auf die Auswahl des Benutzers der angezeigten Optionen. Dies ist ein externer Befehl. Daher können die Parameterbezeichnungen leicht unterschiedlich zu den hier gezeigten sein. Verwenden sie die im Befehl eingebaute Hilfe um mehr über die korrekte Syntax zu erfahren.
CHOICE [/C[:]Tasten] [/N] [/S] [/T[:]c,nn] [Text]

/C[:]Tasten	Angabe der zulässigen Tasten. Standard ist JN.
/N	Keine Anzeige eines ?-Zeichens am Ende der Meldung.
/S	Gross-/Kleinschreibung für Tasten wird beachtet.
/T[:]c,nn	Nach nn Sekunden wird Standardauswahl c ausgeführt.
Text	Meldung, die angezeigt wird.

ERRORLEVEL ist auf die Position der gedrückten Taste aus der Tastenauswahl gesetzt.

SHIFT

Verändert die Position ersetzbarer Parameter in einem Stapelverarbeitungsprogramm.
SHIFT

CALL

Ruft ein Stapelverarbeitungsprogramm von einem anderen aus auf.
CALL [Laufwerk:][Pfad]Dateiname [Parameter]

SET

Setzt oder entfernt MS-DOS-Umgebungsvariablen oder zeigt sie an.
SET [Variable=[Zeichenfolge]]

Variable	Der Name der Umgebungsvariable.
Zeichenfolge	Eine Zeichenfolge, die der Variablen zugewiesen werden soll.

Der Befehl SET ohne Parameter zeigt die derzeitigen Umgebungsvariablen an.

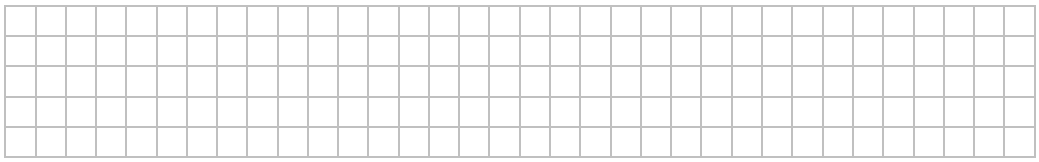
2.3 Internetlinks

http://www.lehrerweb.at/service/online_kurse/win95/skript11.htm	gut
http://www.antonis.de/dos/dos-tuts/morgan/	gut

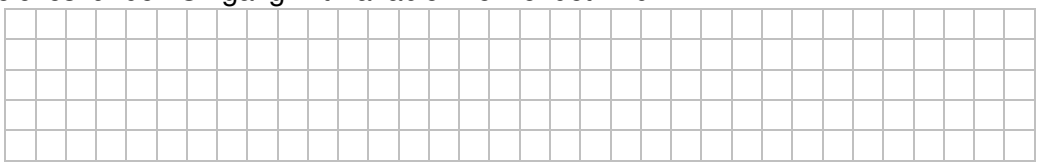
2.4 DOS – Aufgaben

Für folgende Übungen arbeiten wir mit dem CMD-Kommandointerpreter von Windows. Sie können einen neuen Kommandointerpreter über (`START\Ausführen... > CMD.EXE`) starten.

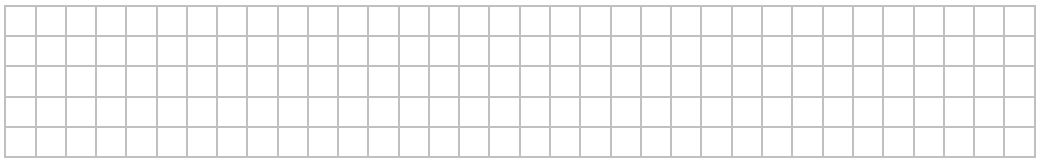
1. Wie können Sie das Laufwerk wechseln? Versuchen Sie z.B. auf das CD-Laufwerk zu wechseln. Wie viele Laufwerke werden auf diese Art unterstützt?



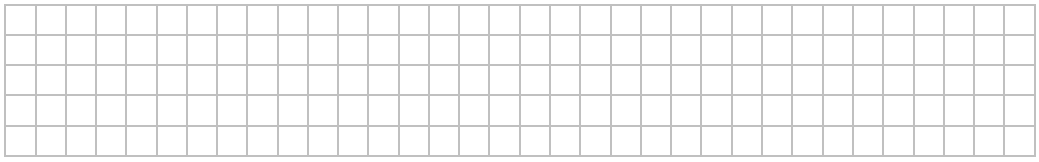
2. Finden Sie heraus, wie Sie Hilfe zu den Befehlen finden – Suchen Sie Hilfe für das Kommando `Set`, welches für den Umgang mit Variablen verwendet wird.



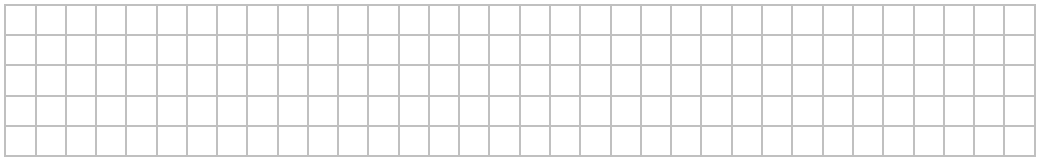
3. Zeigen Sie den Inhalt einer Text-Datei und einer ausführbaren Datei an.



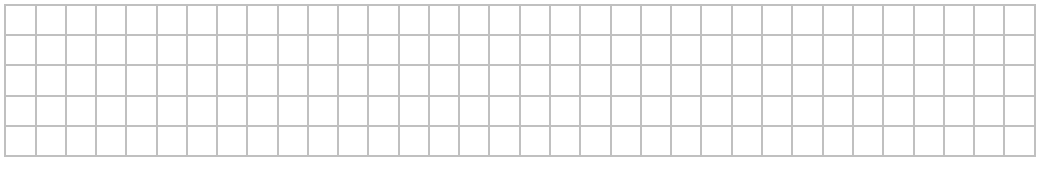
4. Lassen Sie sich den Inhalt des Stammverzeichnis der Partition C in eine Datei Namens ‚Directory.txt' schreiben.



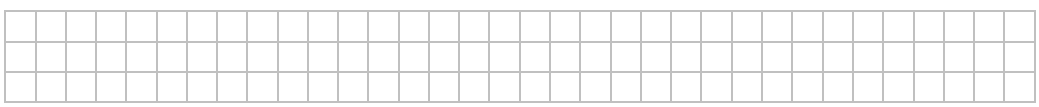
5. Kopieren sie die Datei ‚Directory.txt' in eine Datei Namens ‚temp.txt' mit dem Befehl ‚COPY' Verändern Sie die Datei ‚temp.txt' geringfügig und vergleichen Sie die Datei ‚temp.txt' mit ‚Directory.txt' mit dem Befehl ‚FC'



6. Lassen Sie sich das Inhaltsverzeichnis des Verzeichnisses `C:\WINNT\system32` Seitenweise anzeigen!



7. Lassen Sie sich alle EXE Dateien aus dem Verzeichnis `C:\WINDOWS\system32` auf der Festplatte C: anzeigen.



8. Lassen Sie sich alle Dateien auf Laufwerk `C:\WINDOWS\system32` anzeigen, welche mit dem Buchstaben A beginnt und beliebige weitere Zeichen im Dateipräfix haben.

9. Schreiben Sie ein Batch mit dem Namen `Setup.bat`, welcher zwei Umgebungsvariablen definiert. Der Name der ersten Variablen heisst `DatenPfad` und hat den Wert `C:\Working`. (Legen Sie sich dieses Verzeichnis an!) Die zweite Variable hat den Name `BackupPfad` und den Wert `C:\Backup`.

    a. Prüfen Sie ihren Batch in einem neuen Kommandointerpreter (über Start). Starten Sie aus dem laufenden Kommandointerpreter einen weiteren. Was ist mit den Variablen?

    b. Öffnen Sie wiederum einen neuen Kommandointerpreter (über Start). Starten Sie aus diesem einen weiteren Kommandointerpreter und lassen Sie darin ihren Batch laufen. Prüfen Sie, dass die Variablen gesetzt sind. Beenden Sie anschliessend den Kommandointerpreter (`EXIT`) und prüfen Sie im verbleibenden Kommandointerpreter das vorhanden sein der Variablen. Was ist mit den Variablen?

10. Schreiben Sie einen Batch mit dem Namen `Caller1.bat` und rufen Sie daraus den Batch `Setup.bat` mit Hilfe des Befehls `call` auf. Wie sieht es nun mit den Umgebungsvariablen aus? Sehen Sie sich auch die Hilfe für den Befehl `call` an!

11. Schreiben Sie einen Batch mit dem Namen `Caller2.bat` und rufen Sie daraus den Batch `Setup.bat` mit Hilfe des Befehls `CMD /C` auf. Wie sieht es nun mit den Umgebungsvariablen aus? Sehen Sie sich auch die Hilfe für den Befehl `CMD` an! Was bedeutet der Parameter `/C`?

12. Schreiben Sie einen Batch (`XXCopy.bat`) mit dem Sie ein Verzeichnis mit Inhalt und allen Unterverzeichnissen in ein neues Verzeichnis kopieren können. Das Start- und Ziel-Verzeichnis sollen mit Parameter dem Batch übergeben werden. Probieren Sie den Batch aus. In diesem Zusammenhang prüfen Sie, wie ein Verzeichnis mit gesamtem Inhalt inklusive Unterverzeichnisse gelöscht werden kann. (**Vorsicht!**)

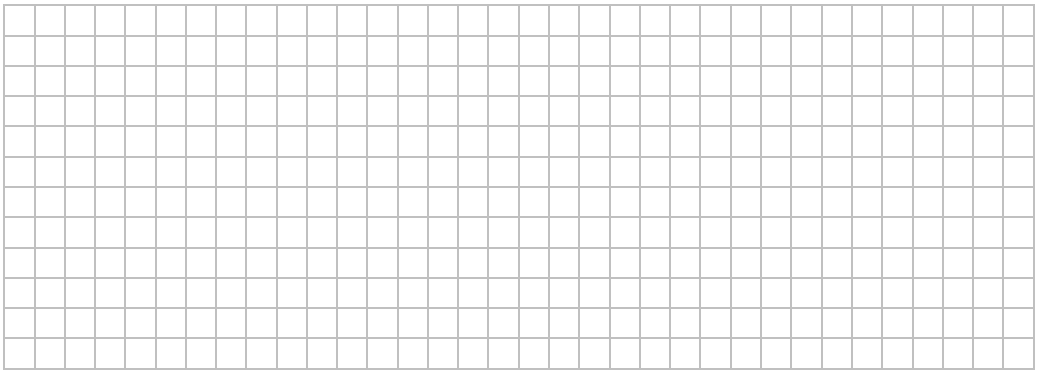
13. Analysieren Sie die Batchdatei `Install.bat` und die von Ihr verwendeten Batchdateien und beschreiben Sie das Verhalten.

14. Schreiben Sie ein kleines Backup Programm (BACKUP.BAT). Das Programm soll 3 Parameter verarbeiten. Der erste Parameter soll die Werte „BACKUP“ und „RESTORE“ verarbeiten. Der zweite Parameter soll das Quell-Verzeichnis aufnehmen und der dritte das Ziel-Verzeichnis aufnehmen. Wird kein Parameter angegeben soll ein kleines Menu erscheinen, in dem BACKUP oder RESTORE angewählt werden kann. Beim starten des Batch soll zuerst ein Batch „BACKUPCONFIG.BAT“ aufgerufen werden, welcher ein vor definiertes Quell- und Ziel-Verzeichnis definiert. Damit müssen der 2. und 3. Parameter beim Aufruf nur optional angegeben werden. Insgesamt sollen folgende Batch geschrieben werden

BACKUP.BAT	Hauptprogramm
BACKUPCONFIG.BAT	Definiert das Standard Ziel- und Quellverzeichnis
BACKUPMENU.BAT	Implementiert ein Menu, Q beendet das Programm.
XXCopy.BAT	Verwenden sie den Batch aus Aufgabe 12

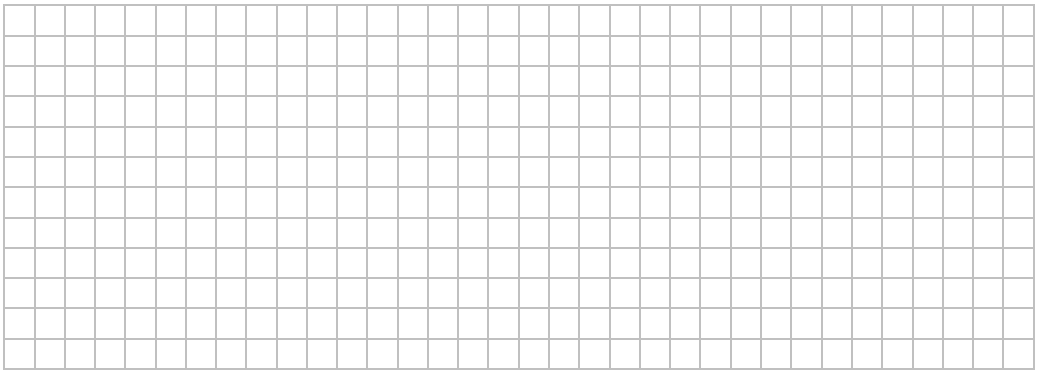
    Vor dem Sichern soll das Quell-Verzeichnis auf vorhanden sein geprüft werden. Fehler sollen dem Benutzer angezeigt werden.

    (Tipp: Sehen Sie sich die Befehl `Choice`, `IF` und `GoTo` an)



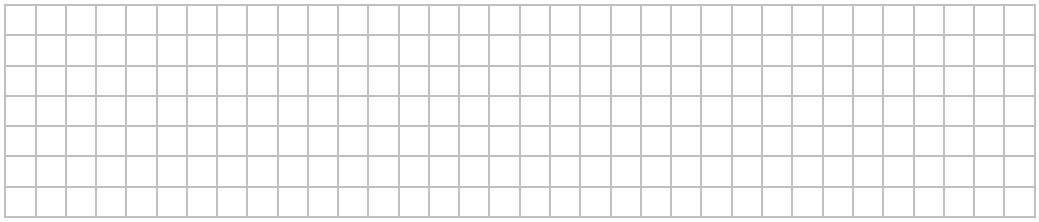
15. Erzeugen Sie auf dem Laufwerk C: im Stammverzeichnis ein Verzeichnis mit dem Namen WORKING. Danach schreiben Sie ein Login Skript mit dem sie den Buchstaben W mit diesem Verzeichnis verbinden können. Binden Sie das Login Skript in das System ein und testen Sie es. Im Explorer sollten sie nach dem Einloggen ein Laufwerk W finden.

    (Tipp: Sehen sie sich den Befehl `SUBST` an)



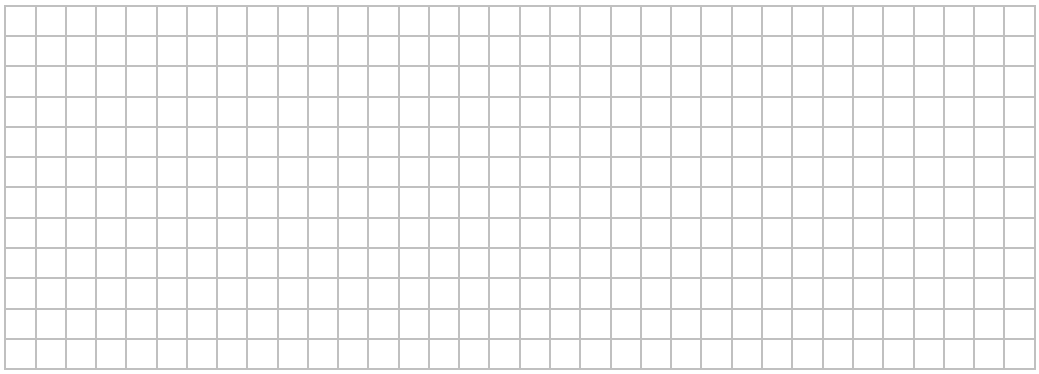
16. Erweitern Sie die Aufgabe 15 dahingehend, dass sie der Computer beim Anmelden mit Ihrem Namen begrüsst. Definieren Sie dazu eine Benutzervariable `HELLO_NAME` unter (START/Systemsteuerung/System) und zeigen sie diese mit dem Laufwerk-Buchstaben an. Der Laufwerksbuchstaben soll auch über eine Benutzervariable definiert sein!

    Bsp.: `Hallo Stephan, das Arbeitslaufwerk ist W`



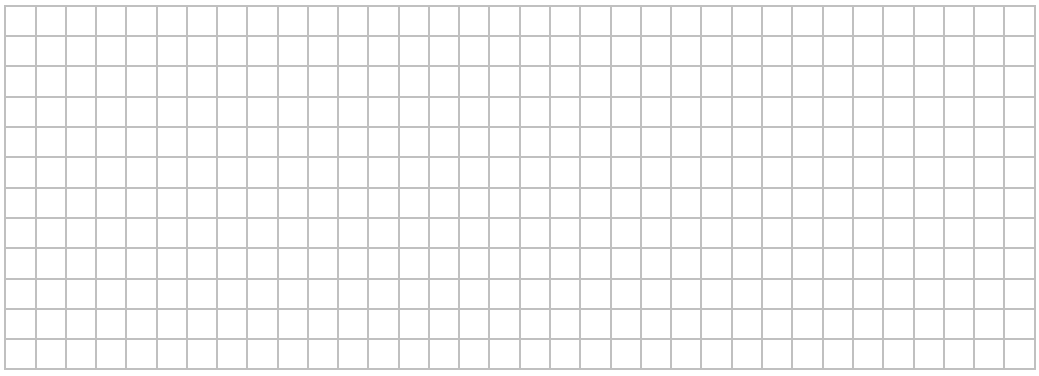
17. Schreiben Sie einen Batch, welcher jeden Tag zu einer bestimmten Zeit einen Backup macht. Verwenden Sie Ihre Lösung von Aufgabe 14. Alles soll in einer Log-Datei festgehalten werden – anhängen!

    (Tipp: Sehen Sie sich den Befehl `AT` an. Beachten Sie dass damit auch Befehle auf anderen Rechnern gestartet werden können, sofern sie über genügend Berechtigungen verfügen.)

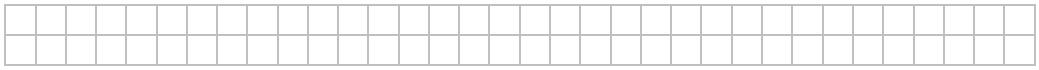


18. Schreiben Sie ein kleines Chat Programm. Beim Start geben sie den Rechnernamen an mit dem sie in Verbindung treten wollen. Ihr Chat-Partner verwendet das selbe Programm. Mit dem Befehl QUIT wird das Programm beendet.

    (Tipp: Sehen Sie sich die Befehle `SET /P` und `NET SEND` an. Der Nachrichtendienst (Messenger) müssen aktiviert und gestartet sein.)



19. Scheiben sie folgende Ausgabe mit dem echo-Befehl: *hallo ich hab's*
    (Escape Zeichen): Wie heisst das Escape-Zeichen?



20. Erforschen sie die Befehle pushd/popd

21. Integrieren sie das Shell-Commando in den Explorer

    Um im Kontextmenü einen Eintrag zum Aufrufen der Shell zu erhalten, gehen Sie wie folgt vor:

    Suchen Sie in der Registrierung unter HKEY_CLASSES_ROOT en Eintrag Verzeichnis und dort den Schlüssel shell.

    Darunter legen Sie nun einen neuen Schlüssel an (der Name ist beliebig, sollte aber selbst dokumentierend sein, beispielsweise cmdShell).

    Darunter legen Sie wieder einen Schlüssel an, der den Namen command tragen muss.

    Im Standardwert für diesen Schlüssel command (der keinen amen trägt) geben Sie nun die auszuführende Befehlszeile an, hier cmd.exe "%1".

    Verlassen Sie nun den Registrierungseditor

22. Ändern sie die Schriftfarbe nach gelb ihrer Commandokonsole:

23. Ändern sie den Titel ihrer Commandokonsole:

24. Leiten sie die Ausgabe der Fehler und der Info in separate Dateien um:

25. Erzeugen sie leere Dateien mit der Commandokonsole! (schwierige Aufgabe)

26. Nummerieren sie die Zeilen einer Textdatei! (schwierige Aufgabe)

27. Kopieren sie die Ausgabe von *ipconfig /all* in die Zwischenablage (nur unter Windows Server 2003 möglich)

28. Hängen sie einer einfachen Textdatei das Programm *Notepad.exe* an und führen sie dieses dann via Textdatei aus (Thema ADS, siehe Internet)

29. Erzeugen sie einen echten Link (Hardwarelink) auf eine beliebige Datei.

30. Finden sie alle Dateien, die jeweils an einem 1. oder 15. eines Monats erstellt wurden.

31. Kopieren Sie alle .log- und .rpt-Dateien aus dem Stammverzeichnis der Rechner von 192.168.1.1 bis 192.168.1.20 in das Verzeichnis logs des Laufwerks C: Ihrer Maschine.

32. Finden Sie alle exe-Dateien in system32, die eine gleichnamige DLL besitzen

2.5 DOS - Befehle (WINDOWS 2000)

ASSOC	Zeigt die Zuordnungen der Dateierweiterungen an oder ändert sie.
AT	Plant die Ausführung von Befehlen und Programmen auf einem Computer.
ATTRIB	Zeigt Dateiattribute an oder ändert sie.
BREAK	Schaltet (zusätzliche) Überwachung für STRG+C ein (ON) oder aus (OFF).
CACLS	Zeigt die Zugriffskontrollisten (ACL) der Dateien an oder ändert sie.
CALL	Ruft ein Stapelverarbeitungsprogramm von einem anderen aus auf.
CD	Wechselt das aktuelle Verzeichnis oder zeigt dessen Namen an.
CHCP	Wechselt die aktuelle Codeseite oder zeigt deren Nummer an.
CHDIR	Wechselt das aktuelle Verzeichnis oder zeigt dessen Namen an.
CHKDSK	Überprüft einen Datenträger und zeigt einen Statusbericht an.
CLS	Löscht den Bildschirminhalt.
CMD	Startet eine neue Instanz des Windows 2000-Befehlsinterpreters.
COLOR	Legt die Standardfarben für den Konsolenhinter- und Vordergrund fest.
COMP	Vergleicht den Inhalt zweier Dateien oder zweier Sätze von Dateien.
COMPACT	Zeigt die Komprimierung der Dateien auf NTFS-Partitionen an oder ändert sie.
CONVERT	Konvertiert FAT-Datenträger in NTFS. Das aktuelle Laufwerk kann nicht konvertiert werden.
COPY	Kopiert eine oder mehrere Dateien an eine andere Position.
DATE	Wechselt das eingestellte Datum oder zeigt es an.
DEL	Löscht eine oder mehrere Dateien.
DIR	Listet die Dateien und Unterverzeichnisse eines Verzeichnisses auf.
DISKCOMP	Vergleicht den Inhalt zweier Disketten.
DISKCOPY	Kopiert den Inhalt einer Diskette auf eine andere Diskette.
DOSKEY	Bearbeitet Befehlseingaben, ruft Befehle zurück und erstellt Makros.
ECHO	Zeigt Meldungen an oder schaltet die Befehlsanzeige ein/aus (ON/OFF).
ENDLOCAL	Beendet die Begrenzung des Gültigkeitsbereiches von Änderungen.
ERASE	Löscht eine oder mehrere Dateien.
EXIT	Beendet den Befehlsinterpreter CMD.EXE.
FC	Vergleicht zwei Dateien oder zwei Sätze von Dateien.
FIND	Sucht in einer oder mehreren Dateien nach einer Zeichenfolge.
FINDSTR	Sucht nach Zeichenketten in Dateien.
FOR	Führt einen Befehl für jede Datei eines Satzes von Dateien aus.
FORMAT	Formatiert einen Datenträger für die Verwendung unter Windows 2000.
FTYPE	Zeigt die Dateitypen an, die bei den Dateierweiterungszuordnungen verwendet werden, oder ändert sie.
GOTO	Setzt die Ausführung eines Stapelverarbeitungsprogramms an einer Marke fort.
GRAFTABL	Ermöglicht Windows 2000, im Grafikmodus einen erweiterten Zeichensatz anzuzeigen.
HELP	Zeigt Hilfe für Windows 2000-Befehle an.
IF	Verarbeitet Ausdrücke mit Bedingungen in einem Stapelverarbeitungsprogramm.
LABEL	Erstellt, ändert oder löscht die Bezeichnung eines Datenträgers.
MD	Erstellt ein Verzeichnis.
MKDIR	Erstellt ein Verzeichnis.

MODE	Konfiguriert Geräte im System.
MORE	Zeigt Daten Seitenweise auf dem Bildschirm an.
MOVE	Verschiebt eine oder mehrere Dateien.
PATH	Legt den Suchpfad für ausführbare Dateien fest oder zeigt diesen an.
PAUSE	Hält die Ausführung einer Stapelverarbeitungsdatei an.
POPD	Wechselt zu dem Verzeichnis, das durch PUSHD gespeichert wurde.
PRINT	Druckt Textdateien während der Verwendung anderer MS-DOS-Befehle.
PROMPT	Modifiziert die Windows 2000-Eingabeaufforderung.
RD	Entfernt (löscht) ein Verzeichnis.
RECOVER	Stellt von einem beschädigten Datenträger lesbare Daten wieder her.
REM	Leitet Kommentare in einer Stapelverarbeitungsdatei oder in der Datei CONFIG.SYS ein.
REN	Benennt eine oder mehrere Dateien um.
RENAME	Benennt eine oder mehrere Dateien um.
REPLACE	Ersetzt Dateien.
RMDIR	Entfernt (löscht) ein Verzeichnis.
SET	Setzt oder entfernt Windows 2000-Umgebungsvariablen oder zeigt sie an.
SETLOCAL	Startet die Begrenzung des Gültigkeitsbereiches von Änderungen.
SHIFT	Verändert die Position ersetzbarer Parameter in einem Stapelverarbeitungsprogramm.
SORT	Gibt Eingabe sortiert auf Bildschirm, Datei oder anderes Gerät aus.
SUBST	Weist einem Pfad eine Laufwerksbezeichnung zu.
START	Startet ein eigenes Fenster, um das Programm auszuführen.
TIME	Stellt die Systemzeit ein oder zeigt sie an.
TREE	Zeigt die Verzeichnisstruktur eines Laufwerks oder Pfads grafisch an.
TYPE	Zeigt den Inhalt einer Textdatei an.
VER	Zeigt die Nummer der verwendeten Windows 2000-Version an.
VERIFY	Legt fest, ob MS-DOS überwachen soll, dass Dateien korrekt auf Datenträger geschrieben werden.
VOL	Zeigt die Bezeichnung und Seriennummer eines Datenträgers an.
XCOPY	Kopiert Dateien und Verzeichnisstrukturen.