

## Batch Programmierung



Datum: 19.10.12

Autor: Kesselmark, Pascal



KlingelInberg GmbH  
Peterstraße 45  
D-42499 Hückeswagen  
Fon: 0049-2192-810  
Fax: 0049-2192-81200

[info@klingelInberg.com](mailto:info@klingelInberg.com)  
[www.klingelInberg.com](http://www.klingelInberg.com)

Amtsgericht Köln HRB 37950  
Ust-IdNr.: DE 184637081  
Steuernummer: 221/5749/0125

KlingelInberg AG  
Binzmühlestrasse 171  
CH-8050 Zürich  
Fon: 0041-44-278 78 79  
Fax: 0041-44-273 15 56

© 2012 KlingelInberg AG. Alle Rechte vorbehalten.

Das Urheberrecht dieser Dokumentation bleibt bei der KlingelInberg AG. Die Dokumentation enthält Informationen technischer Art, die weder vollständig noch teilweise kopiert, verbreitet oder zu Zwecken des Wettbewerbs unbefugt verwertet oder anderen mitgeteilt werden dürfen.

# Inhaltsverzeichnis

<b>1</b>	<b>Allgemeine Information</b>	<b>5</b>
1.1	Batch-Programmierung	5
1.2	Allgemeine Informationen	6
<b>2</b>	<b>Workshop</b>	<b>7</b>
2.1	Ausgabe	7
2.1.1	Erste Ausgabe erstellen.....	7
2.1.2	Ausgabe anpassen .....	9
2.1.3	Kommentar hinzufügen.....	9
2.1.4	Titel anzeigen .....	10
2.1.5	Systemvariablen anzeigen.....	11
2.2	Eingabe	12
2.2.1	Variable definieren und ausgeben .....	12
2.2.2	Eingabe übernehmen und ausgeben .....	12
2.2.3	Rechnen mit eingegebenen Werten .....	13
2.3	Abfrage	13
2.3.1	Abfrage einer Zahl .....	14
2.3.2	Abfrage eines Buchstaben.....	15
2.4	Sprungmarke	15
2.4.1	Sprungmarke aufrufen .....	16
2.4.2	Parameter übergeben.....	17
2.5	Verzeichnisse und Dateien	18
2.5.1	Verzeichnis anlegen .....	18
2.5.2	Verzeichnis unter spezifischem Pfad erstellen .....	19
2.5.3	Verzeichnispfad aus dynamischen Inhalten erstellen .....	19
2.5.4	Datei kopieren.....	20
2.5.5	Dateien umbenennen .....	21
2.5.6	Textdatei erstellen und befüllen.....	22
<b>3</b>	<b>Anwendungsbeispiele</b>	<b>24</b>
3.1	Logdatei erstellen	24

3.2	TEMP-Verzeichnis löschen	25
<b>4</b>	<b>Referenz</b>	<b>26</b>
4.1	Batch-Befehle	26
4.1.1	@ (at-Zeichen) .....	26
4.1.2	: (Doppelpunkt) .....	27
4.1.3	CALL .....	28
4.1.4	CLS 30	
4.1.5	COLOR .....	31
4.1.6	ECHO .....	31
4.1.7	FOR 32	
4.1.8	GOTO .....	34
4.1.9	IF 35	
4.1.10	PAUSE .....	37
4.1.11	PUSHD / POPD .....	39
4.1.12	REM 39	
4.1.13	START .....	39
4.1.14	Variablen .....	41

# 1 Allgemeine Information

## 1.1 Batch-Programmierung

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

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

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

### Wie erstelle ich eine Batchdatei?

Wie bereits gesagt, reicht ein einfacher Editor aus. Empfehlen kann man guten Gewissens den quelloffenen und kostenlosen Notepad++-Editor, als bessere Alternative zu MS Notepad. Speichern Sie dann die Datei mit der Endung \*.bat ab. Anschließend müssen Sie nur die Datei starten: Dies können Sie entweder direkt in Windows oder indem Sie in der DOS-Eingabeaufforderung in das entsprechende Verzeichnis wechseln und den Dateinamen eingeben.

Unter Windows NT-kompatiblen Betriebssystemen gibt es seit Windows 2000 auch Batchdateien mit der Endung \*.cmd. Diese werden genau wie Batchdateien mit der Endung \*.bat verarbeitet bzw. ausgeführt. Die Endung \*.cmd wurde ursprünglich aus Kompatibilitätsgründen zu OS/2, einem Betriebssystem das MS ursprünglich in Kooperation mit IBM entwickelte, mit Windows NT 3.x eingeführt.

## 1.2 Allgemeine Informationen

Batch-Programmierung: <http://de.wikibooks.org/wiki/Batch-Programmierung>

Allgemeine Informationen: <http://de.wikipedia.org/wiki/Stapelverarbeitung>

Infos zur cmd.exe: <http://de.wikipedia.org/wiki/cmd.exe>

## 2 Workshop

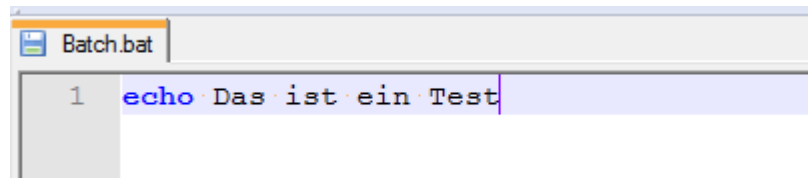
### 2.1 Ausgabe

In diesem Abschnitt wird die Bildschirmausgabe der Batch-Datei erstellt.

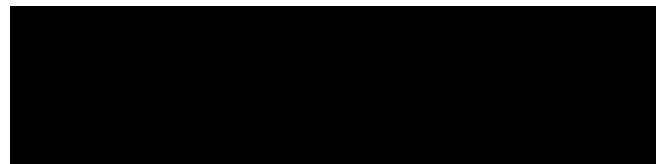
#### WICHTIG:

Prinzipiell haben Batch-Dateien ein Problem mit Sonderzeichen. Alles ausserhalb des ASCII-Zeichensatzes sollte vermieden werden (ä, ö, è, ß).

#### 2.1.1 Erste Ausgabe erstellen



1. Im Texteditor eine neue Datei erstellen.
2. "echo Das ist ein Test" eingeben.
3. Batch-Datei speichern
4. Batch-Datei ausführen.

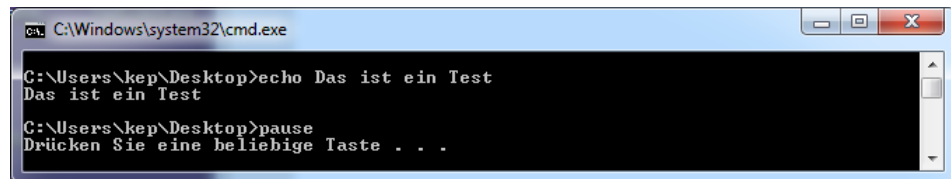


Das Fenster wird nach der Ausführung leider sofort wieder geschlossen.

Dagegen muss man etwas machen. Wir fügen am Schluss eine Pause ein, die das Drücken einer Taste erzwingt.



1. "pause" am Schluss einfügen.
2. Batch-Datei speichern.
3. Batch-Datei ausführen.



```
C:\Windows\system32\cmd.exe

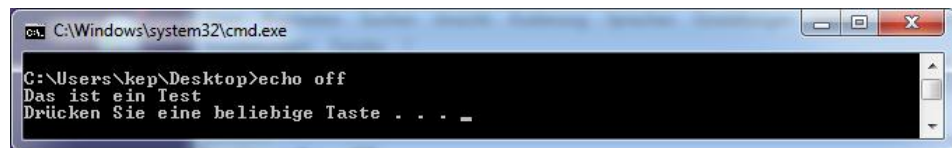
C:\Users\kep\Desktop>echo Das ist ein Test
Das ist ein Test

C:\Users\kep\Desktop>pause
Drücken Sie eine beliebige Taste . . .
```

Jetzt wird der Text ausgegeben, jedoch doppelt und auch noch mit der Befehlszeile. Das wollen wir ändern. Indem wir "echo off" am Anfang hinzufügen.

```
1 echo off
2 echo Das ist ein Test
3
4 pause
```

1. "echo off" am Anfang einfügen.
2. Batch-Datei speichern.
3. Batch-Datei ausführen.



```
C:\Windows\system32\cmd.exe

C:\Users\kep\Desktop>echo off
Das ist ein Test


C:\Users\kep\Desktop>pause
Drücken Sie eine beliebige Taste . . .
```

Schon besser, jetzt wird nur noch beim ersten Eintrag (echo off) die Befehlszeile angezeigt, da das Programm erst ab der zweiten Zeile "echo off" ausführt.

Aus diesem Grund muss die erste Zeile mit "@" ausgeblendet werden.

```
1 @echo off
2 echo Das ist ein Test
3
4 pause
```

1. "@" vor "echo off" einfügen --> @echo off.
2. Batch-Datei speichern.
3. Batch-Datei ausführen.



```
C:\Windows\system32\cmd.exe

C:\Users\kep\Desktop>@echo off
Das ist ein Test

C:\Users\kep\Desktop>pause
Drücken Sie eine beliebige Taste . . .
```

Jetzt wird nur noch der Text aus der Batch-Datei angezeigt.



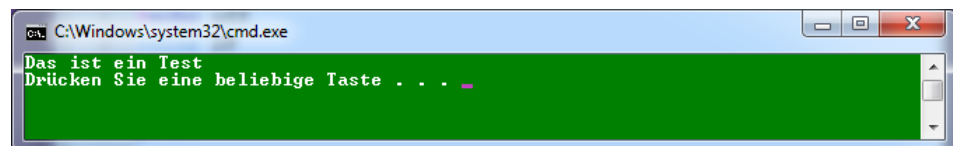
## 2.1.2 Ausgabe anpassen

```

1 @echo off
2 color 2F
3 echo Das ist ein Test
4
5 pause

```

1. Farbe ändern: "color 2F" hinzufügen = grüner Hintergrund und weisse Schrift.
2. Batch-Datei speichern.
3. Batch-Datei ausführen.



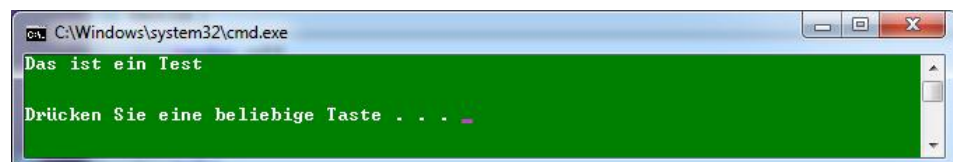
Jetzt noch Leerzeilen zwischen dem Text und der Mitteilung hinzufügen.

```

1 @echo off
2 color 2F
3 echo Das ist ein Test
4 echo.
5 echo.
6
7 pause

```

1. "echo." hinzufügen. Da am Anfang "@echo off" steht, muss bei "echo." ein Punkt am Schluss stehen, sonst kommt eine Fehlermeldung.
2. Batch-Datei speichern.
3. Batch-Datei ausführen.



## 2.1.3 Kommentar hinzufügen

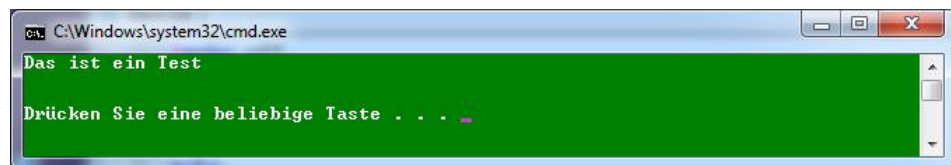
Beim Programmieren ist es hilfreich, wenn man den Code kommentieren kann, um ihn später wieder einfacher zu verstehen, oder wenn eine andere Person damit arbeiten muss.

Dafür wird der Befehl "REM" (Remark) verwendet.



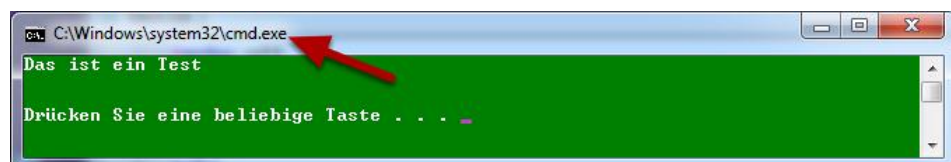
```
1 @echo off
2 REM *****
3 REM *** Test-Batch-Datei *****
4 REM *****
5 REM *** Erstellt von Pascal Kesselmark *****
6 REM *** Version 1.0.0 - 24.10.2012 *****
7 REM *****
8 color 2F
9 echo Das ist ein Test
10 echo.
11 echo.
12
13 pause
```

1. Kommentar einfügen. Zeile beginnt immer mit REM und einem Leerschlag.
2. Batch-Datei speichern.
3. Batch-Datei ausführen.



Die Kommentare werden nicht angezeigt. Wichtig ist aber, dass zuerst "@echo off" stehen muss!

#### 2.1.4 Titel anzeigen



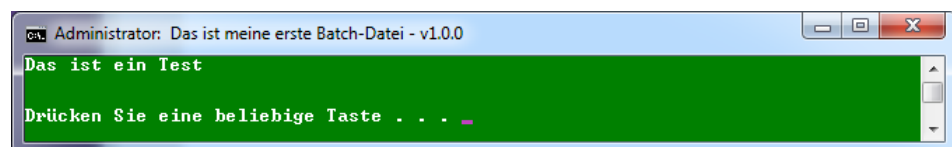
Standardmässig zeigt die Batch-Datei in der Titelzeile den Pfad der Anwendung an. Dies sieht nicht schön aus und ist bei einem Support-Fall auch nicht hilfreich. Besser wäre den Programmnamen und den Titel anzuzeigen.

```

1 @echo off
2 REM *****
3 REM *** Test-Batch-Datei ***
4 REM *****
5 REM *** Erstellt von Pascal Kesselmark ***
6 REM *** Version 1.0.0 - 24.10.2012 ***
7 REM *****
8 color 2F
9 title Das ist meine erste Batch-Datei - v1.0.0
10 echo Das ist ein Test
11 echo.
12 echo.
13
14 pause

```

1. "title Das ist meine erste Batch-Datei - v1.0.0"
2. Batch-Datei speichern.
3. Batch-Datei ausführen.



Der Titel wird nun angezeigt.

## 2.1.5 Systemvariablen anzeigen

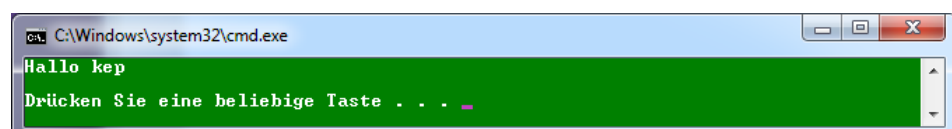
Man kann natürlich auch noch Systemvariablen anzeigen. Wir sprechen nun den Benutzer mit Namen an. Dafür verwenden wir die Umgebungsvariable %USERNAME%.

```

1 @echo off
2 color 2F
3
4 echo Hallo %USERNAME%
5 echo.
6
7 pause

```

1. "echo Hallo %USERNAME%" eingeben.
2. Batch-Datei speichern.
3. Batch-Datei ausführen.




## 2.2 Eingabe

Nachdem die Bildschirmausgabe abgehandelt wurde, nun zum nächsten Punkt die Eingabe und deren Verwertung.

### 2.2.1 Variable definieren und ausgeben

```
1 @echo off
2 color 2F
3
4 set PFAD= \\srv1\Daten\Projekte\Auftraege
5
6 echo Installationspfad: %PFAD%
7 echo.
8 pause
```

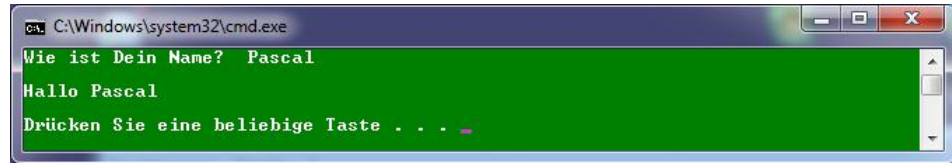
1. Eingabe "set PFAD= \\srv1\Daten\Projekte\Auftraege " eingeben.
2. Ausgabe: "echo Installationspfad: %PFAD%" eingeben.
3. Batch-Datei speichern.
4. Batch-Datei ausführen.



### 2.2.2 Eingabe übernehmen und ausgeben

```
1 @echo off
2 color 2F
3
4 set /p NAME= Wie ist Dein Name? . .
5 echo.
6 echo Hallo %NAME%
7 echo.
8
9 pause
```

1. Eingabe "set /p NAME= Wie ist Dein Name? " eingeben.
2. Ausgabe: "echo Hallo %NAME%" eingeben.
3. Batch-Datei speichern.
4. Batch-Datei ausführen.



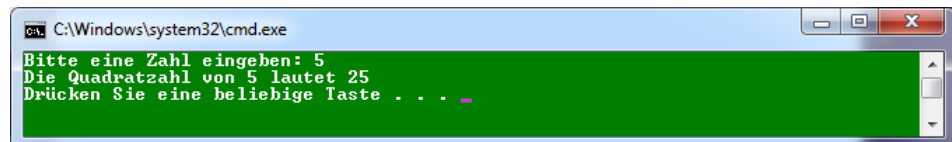
```
C:\Windows\system32\cmd.exe
Wie ist Dein Name? Pascal
Hallo Pascal
Drücken Sie eine beliebige Taste . . .
```

**Speziell:** Das "set /p" schaltet den Befehl auf den Eingabe-Modus um.

### 2.2.3 Rechnen mit eingegebenen Werten

```
1 @echo off
2 color 2F
3
4 set /p ZAHL= Bitte eine ganze Zahl eingeben:
5 set /a Resultat=%Zahl%*%Zahl%
6 echo Die Quadratzahl von %ZAHL% lautet %Resultat%
7
8 pause
```

1. "set /p ZAHL= Bitte eine ganze Zahl eingeben: " eingeben.
2. "set /a Resultat=%Zahl%\*%Zahl%" eingeben.
3. "echo Die Quadratzahl von %ZAHL% lautet %Resultat%" eingeben.
4. Batch-Datei speichern.
5. Batch-Datei ausführen.



```
C:\Windows\system32\cmd.exe
Bitte eine Zahl eingeben: 5
Die Quadratzahl von 5 lautet 25
Drücken Sie eine beliebige Taste . . .
```

## 2.3 Abfrage

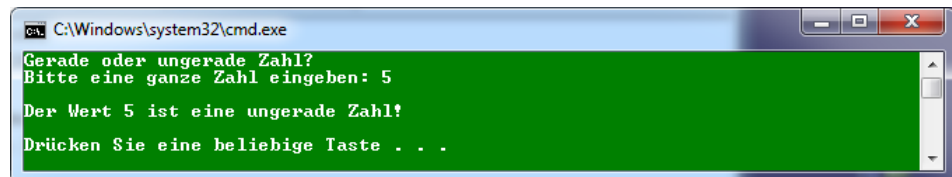
Eingaben müssen/sollen auch ausgewertet werden. Dafür gibt es die Abfragen im Batch-File.

Mit der IF-Abfrage wird geprüft, ob eine Variable einem geprüften Wert entspricht. Wenn diese Bedingung erfüllt ist (TRUE), wird der Befehl in der nächsten Klammer ausgeführt. Falls die Bedingung nicht erfüllt wird (FALSE), wird der Befehl in der zweiten Klammer (nach ELSE) ausgeführt. ELSE und die zweite Klammer können auch weggelassen werden.

### 2.3.1 Abfrage einer Zahl

```
1 @echo off
2 color 2F
3
4 echo Gerade oder ungerade Zahl?
5 set /p ZAHL= Bitte eine ganze Zahl eingeben:
6 echo.
7 set /a Resultat=%ZAHL% %% 2
8
9 IF "%Resultat%" == "0" (echo Der Wert %ZAHL% ist eine gerade
Zahl!) ELSE (echo Der Wert %ZAHL% ist eine ungerade Zahl!)
10 echo.
11 pause
```

1. "set /p ZAHL= Bitte eine ganze Zahl eingeben: " eingeben.
2. "set /a Resultat=%ZAHL% %% 2" eingeben.
3. "IF "%Resultat%" == "0" (echo Der Wert %ZAHL% ist eine gerade Zahl!) ELSE (echo Der Wert %ZAHL% ist eine ungerade Zahl!)" eingeben.
4. Batch-Datei speichern.
5. Batch-Datei ausführen.



**Speziell:** "set /a" schaltet den Arithmetik-Modus ein. "Resultat=%ZAHL% %% 2" bedeutet Modulo 2 von ZAHL. Modulo berechnet den Rest (Resultat) der Division "ZAHL" geteilt durch "2". So kann man in diesem Fall feststellen, ob es sich um eine gerade und ungerade Zahl handelt.

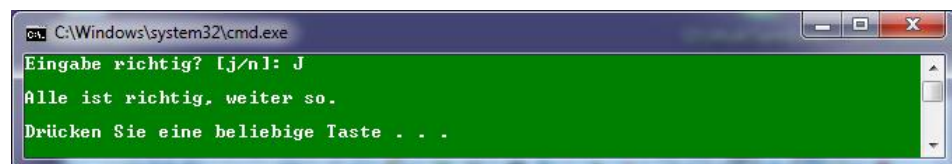
## 2.3.2 Abfrage eines Buchstaben

```

1 @echo off
2 color 2F
3
4 set /p OK= Eingabe richtig? [j/n]:
5 echo.
6
7 IF /I "%OK%" == "j" (echo Alle ist richtig, weiter so.) ELSE
  (echo hier koennte man das Programm abrechen)
8 echo.
9 pause

```

1. "set /p OK= Eingabe richtig? [j/n]: " eingeben.
2. "IF /I "%OK%" == "j" (echo Alle ist richtig, weiter so.) ELSE (echo hier koennte man das Programm abrechen)" eingeben.
3. Batch-Datei speichern.
4. Batch-Datei ausführen.



**Speziell:** Der Befehl "IF /I" macht, dass bei der Eingabe die Gross-/Kleinschreibung ignoriert wird.

## 2.4 Sprungmarke

Mit Sprungmarken kommen wir dem objektorientierten Programmieren immer näher. Es sind Routinen die entweder aus programmtechnischen Gründen "gekapselt" werden sollen oder dazu führen weniger Redundanzen zu haben.

Der Befehl **CALL :NAME** ruft die entsprechende Subroutine auf und kehrt danach auf die folgende Zeile zurück (nur wenn "GOTO :eof" verwendet wird!)



## 2.4.1 Sprungmarke aufrufen

```

Batch.bat | _make_new_masch.bat | _make_new_MLK.bat |
1  @echo off
2  color 2F
3
4  set /p OK= Eingabe richtig? [j/n]:
5  echo.
6  IF /I "%OK%" == "j" (CALL :JA) ELSE (CALL :NEIN)
7  echo Hier ist der naechste Text
8  pause
9
10 REM *** Subroutinen ***
11 :JA
12 echo Alles ist richtig, weiter so.
13 GOTO :eof
14
15 :NEIN
16 echo Hier wird das Programm abgebrochen.
17 GOTO :eof

```

Folgendes eingeben:

set /p OK= Eingabe richtig? [j/n]:

echo.

IF /I "%OK%" == "j" (CALL :JA) ELSE (CALL :NEIN)

echo Hier ist der naechste Text

pause

REM \*\*\* Subroutinen \*\*\*

:JA

echo Alles ist richtig, weiter so.

GOTO :eof

:NEIN

echo Hier wird das Programm abgebrochen.

GOTO :eof



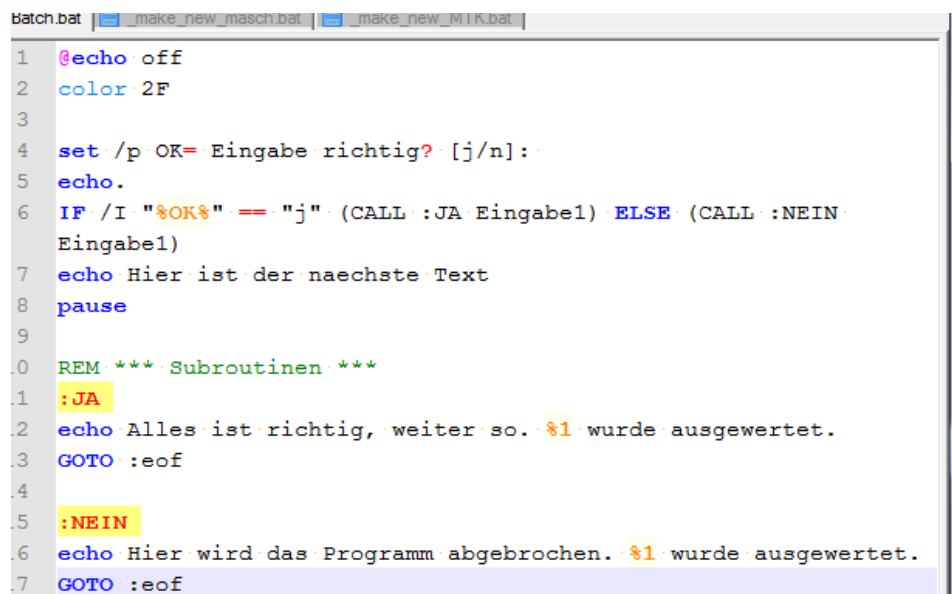


```

C:\Windows\system32\cmd.exe
Eingabe richtig? [j/n]: j
Alles ist richtig, weiter so.
Hier ist der naechste Text
Drücken Sie eine beliebige Taste . . .
  
```

**Speziell:** Der Befehl "GOTO :eof" beendet die Subroutine (CALL) und kehrt zum Ausgangspunkt zurück (nächste Zeile).

## 2.4.2 Parameter übergeben



```

Batch.bat  _make_new_masch.bat  _make_new_MIK.bat
1  @echo off
2  color 2F
3
4  set /p OK= Eingabe richtig? [j/n]:
5  echo.
6  IF /I "%OK%" == "j" (CALL :JA Eingabe1) ELSE (CALL :NEIN
   Eingabe1)
7  echo Hier ist der naechste Text
8  pause
9
10 REM *** Subroutinen ***
11 :JA
12 echo Alles ist richtig, weiter so. %1 wurde ausgewertet.
13 GOTO :eof
14
15 :NEIN
16 echo Hier wird das Programm abgebrochen. %1 wurde ausgewertet.
17 GOTO :eof
  
```

Folgendes eingeben:

set /p OK= Eingabe richtig? [j/n]:

echo.

IF /I "%OK%" == "j" (CALL :JA Eingabe1) ELSE (CALL :NEIN Eingabe1)

echo Hier ist der naechste Text

pause

REM \*\*\* Subroutinen \*\*\*

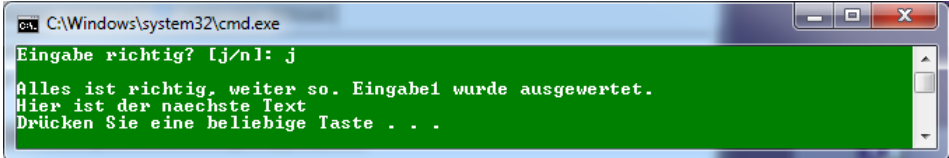
:JA

echo Alles ist richtig, weiter so. %1 wurde ausgewertet.

GOTO :eof

:NEIN

echo Hier wird das Programm abgebrochen. %1 wurde ausgewertet.  
GOTO :eof



**Speziell:** Einem Subrutinenaufwurf kann ein Wert oder eine Variable übergeben werden. Diese kann dann in der Subroutine weiterverarbeitet werden.

## 2.5 Verzeichnisse und Dateien

Mit einer Batch-Datei kann man auch DOS-Befehle ausführen. Diese Befehle benötigen wir, um Verzeichnisse anzulegen, Dateien zu kopieren oder umzubenennen.

### 2.5.1 Verzeichnis anlegen

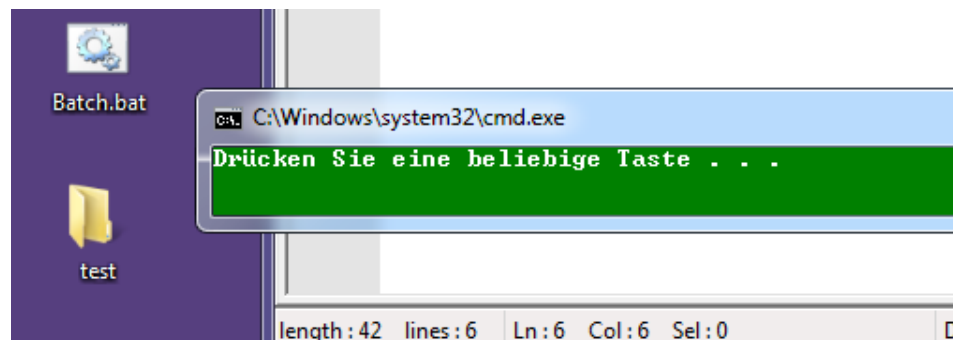
Der Befehl mkdir (Make Directory) erstellt ein Verzeichnis.

Beachte: Das Verzeichnis, falls die explizite Pfadangabe fehlt, wird immer am Ausführungsort der Batch-Datei erstellt.

```
1 @echo off
2 color 2F
3
4 mkdir test
5
6 pause
```

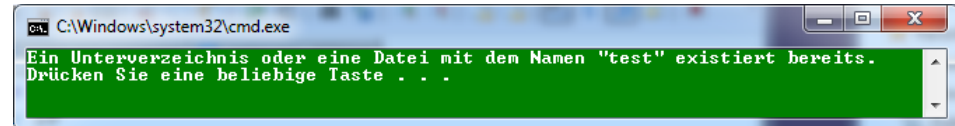
Folgendes eingeben:

mkdir test



Das Verzeichnis "test" wurde erstellt.

Falls das Verzeichnis schon existiert, wird lediglich eine Fehlermeldung generiert und das Programm läuft weiter.

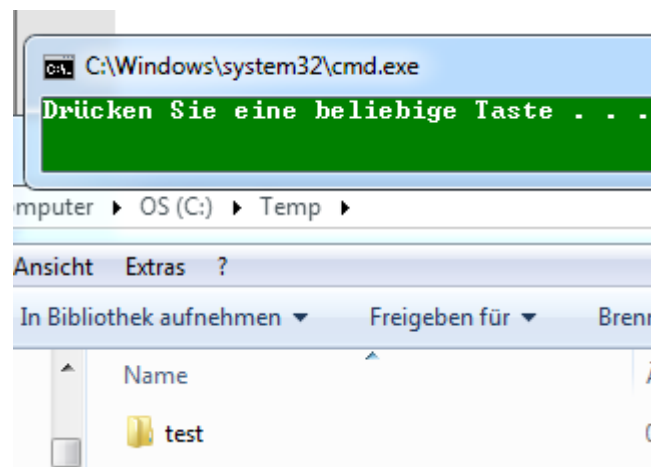


## 2.5.2 Verzeichnis unter spezifischem Pfad erstellen

```
1 @echo off
2 color 2F
3
4 mkdir C:\temp\test
5
6 pause
```

Folgendes eingeben:

mkdir C:\temp\test



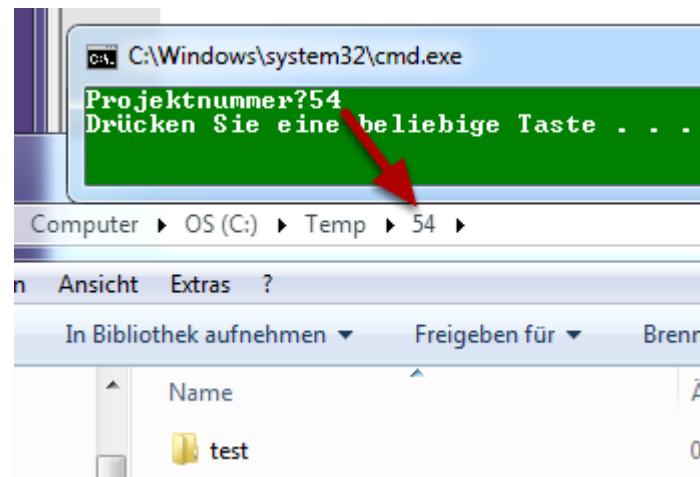
## 2.5.3 Verzeichnispfad aus dynamischen Inhalten erstellen

```
1 @echo off
2 color 2F
3
4 set /p PNUM=Projektnummer?
5 mkdir C:\temp\%PNUM%\test
6
7 pause
```

Folgendes eingeben:

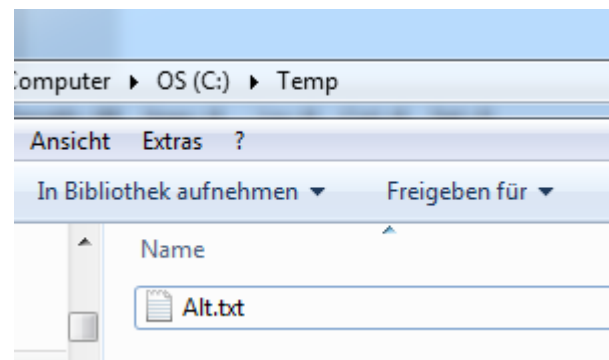
```
set /p PNUM=Projektnummer?  
mkdir C:\temp\%PNUM%\test
```

Die Variable PNUM wird mit der Abfrage gefüllt und danach wird die Variable in den Dateipfad "eingebaut".



## 2.5.4 Datei kopieren

Dateien zu kopieren ist mit DOS-Befehlen eine einfach zu lösende Aufgabe. Man kann die neue Datei umbenennen und sogar die Erweiterung ändern.



Im Verzeichnis C:/temp die Datei "Alt.txt" anlegen.

```
1 @echo off  
2 color 2F  
3  
4 COPY C:\temp\Alt.txt C:\temp\Neu.txt  
5  
6 pause
```

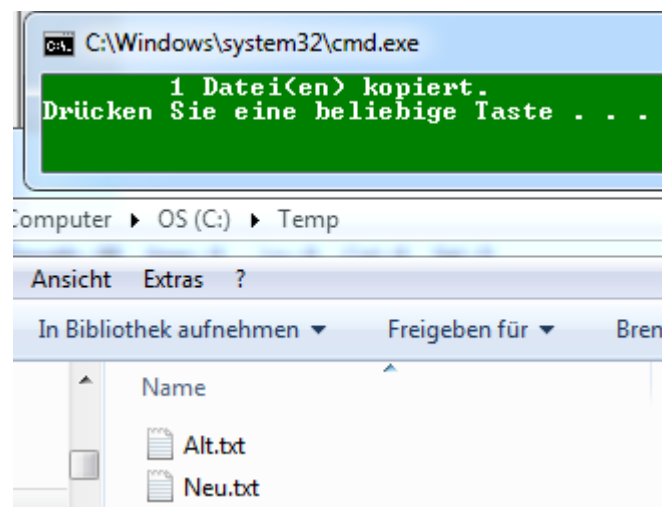
Folgendes eingeben:

COPY C:\temp\Alt.txt C:\temp\Neu.txt

### Wichtig:

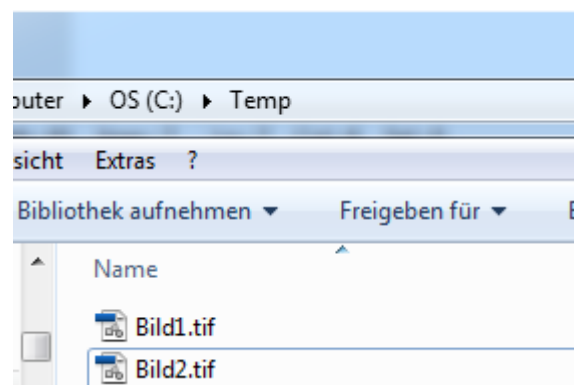
Datei- und Pfadnamen mit Sonderzeichen (Leerzeichen sowie {}, ^, !, +, ~) müssen beim Copy-Befehl und den meisten anderen Befehlen in doppelten Anführungszeichen (") angegeben werden, zum Beispiel:

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



## 2.5.5 Dateien umbenennen

Eine Funktion, die man braucht, wenn z.B. das führende Programm eine Dateierweiterung verwendet, die das nachfolgende Programm nicht erkennt. Z.B. TIF und TIFF.

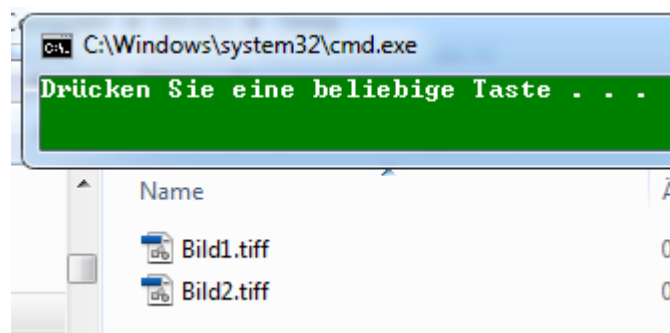


```
1 @echo off
2 color 2F
3
4 cd C:\temp\
5 ren *.tif *.tiff
6
7 pause
```

Folgendes eingeben:

```
cd C:\temp\
ren *.tif *.tiff
```

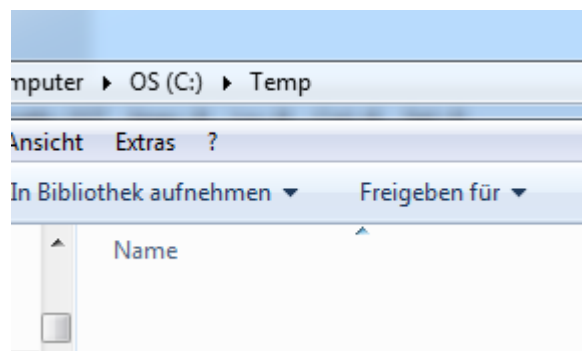
Mit cd (Change Directory) wird in ein Verzeichnis gewechselt. Der Befehl ren (Rename) benennt die Dateien um.



## 2.5.6 Textdatei erstellen und befüllen

Ob Steuerdatei oder Logfile. Mit einer Batch-Datei kann man auch Textdateien erstellen. Prinzipiell ist es wie eine Bildschirmausgabe, die aber in eine Datei umgeleitet wird.

Falls die Datei nicht besteht, wird diese angelegt. Falls die Datei besteht, wird der Eintrag auf die nächste Zeile geschrieben.



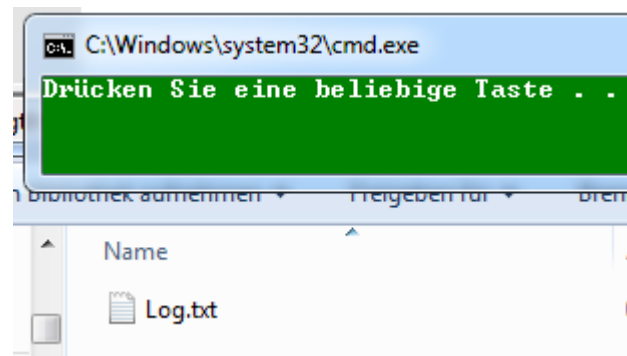
Das Verzeichnis ist leer.

```
1 @echo off
2 color 2F
3
4 cd C:\temp\
5 echo Dies ist ein Eintrag >>Log.txt
6 echo ----- >>Log.txt
7
8 pause
```

Folgendes eingeben:

```
cd C:\temp\
echo Dies ist ein Eintrag >>Log.txt
echo ----- >>Log.txt
```

Der Befehl ">>" schreibt die Ausgabe in die definierte Datei.



Der Inhalt der Log.txt sieht wie folgt aus:

```
1 Dies ist ein Eintrag
2 -----
3
```

Und so sieht der Inhalt aus, nachdem der Batch-File 3x ausgeführt wurde.

```
1 Dies ist ein Eintrag
2 -----
3 Dies ist ein Eintrag
4 -----
5 Dies ist ein Eintrag
6 -----
7
```



## 3 Anwendungsbeispiele

### 3.1 Logdatei erstellen

#### Anwendung:

Erstellen einer zentralen Logdatei, um zu sehen, wer, was, wann mit der Batch-Datei gemacht hat.

Man kann diese Logfunktion aber auch verwenden, um Zustands- oder Fehlermeldungen zu protokollieren.

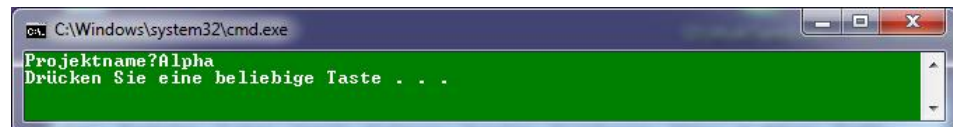
```
1 @echo off
2 color 2F
3
4 set /p PNAME= Projektname?
5 cd C:\temp\
6 echo %date:~0% - %time:~0,8% Uhr - Projektname: %PNAME% -
  Ersteller: %USERNAME%>>Log.txt
7
8 pause
```

Folgendes eingeben:

set /p PNAME= Projektname?

cd C:\temp\

echo %date:~0% - %time:~0,8% Uhr - Projektname: %PNAME% - Ersteller:  
%USERNAME%>>Log.txt



```
1 09.10.2012 - 11:06:21 Uhr - Projektname: Alpha - Ersteller: kep
2
```

```
1 09.10.2012 - 11:06:21 Uhr - Projektname: Alpha - Ersteller: kep
2 09.10.2012 - 11:07:13 Uhr - Projektname: Beta - Ersteller: kep
3 09.10.2012 - 11:07:22 Uhr - Projektname: Alpha - Ersteller: kep
4 09.10.2012 - 11:07:27 Uhr - Projektname: Charly - Ersteller: kep
5 09.10.2012 - 11:07:32 Uhr - Projektname: Delta - Ersteller: kep
6 09.10.2012 - 11:07:38 Uhr - Projektname: Foxtrott - Ersteller: kep
7 09.10.2012 - 11:07:48 Uhr - Projektname: Big Mac - Ersteller: kep
8
```



## 3.2 TEMP-Verzeichnis löschen

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

```
1 Echo off
2 del /f /s /q "%HOMEPATH%\Lokale Einstellungen\Temp\*.*"
3 del /f /s /q "%HOMEPATH%\Lokale Einstellungen\Temporary
  Internet Files\*.*"
4 del /f /s /q "%HOMEPATH%\Lokale Einstellungen\Verlauf\*.*"
5 del /f /s /q "%windir%\Temp\*.*"
6 del /f /s /q "%windir%\Prefetch\*.*"
7 del /f /s /q "%windir%\Temp\*.*"
8 rmdir /s /q "%HOMEPATH%\Lokale Einstellungen\Temp\"
9 rmdir /s /q "%windir%\Prefetch\"
10
11 diskperf -n
12 ipconfig /flushdns
```

Echo off

del /f /s /q "%HOMEPATH%\Lokale Einstellungen\Temp\\*.\*"

del /f /s /q "%HOMEPATH%\Lokale Einstellungen\Temporary Internet Files\\*.\*"

del /f /s /q "%HOMEPATH%\Lokale Einstellungen\Verlauf\\*.\*"

del /f /s /q "%windir%\Temp\\*.\*"

del /f /s /q "%windir%\Prefetch\\*.\*"

del /f /s /q "%windir%\Temp\\*.\*"

rmdir /s /q "%HOMEPATH%\Lokale Einstellungen\Temp\"

rmdir /s /q "%windir%\Prefetch\"

diskperf -n

ipconfig /flushdns

## 4 Referenz

### 4.1 Batch-Befehle

#### Achtung Leerzeichen!

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

#### 4.1.1 @ (at-Zeichen)

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

#### Syntax:

@befehl

#### Inhalt:

```
1 echo Diese Zeile wird mit Befehlszeile ausgeführt...
2 @echo und diese ohne!
3 pause
```

#### Ausgabe:



```
C:\Windows\system32\cmd.exe


C:\Users\kep\Desktop>echo Diese Zeile wird mit Befehlszeile ausgeführt...
Diese Zeile wird mit Befehlszeile ausgeführt...
und diese ohne!

C:\Users\kep\Desktop>pause
Drücken Sie eine beliebige Taste . . . _
```

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

```
1 @echo off
2 REM Verhindert, dass dieser Kommentar angezeigt wird.
3 pause
```

#### Ohne @echo off:



```

C:\Windows\system32\cmd.exe
C:\Users\kep\Desktop>REM Verhindert, dass dieser Kommentar angezeigt wird.
C:\Users\kep\Desktop>pause
Drücken Sie eine beliebige Taste . . .
  
```

### 4.1.2 : (Doppelpunkt)

Sprungmarke für ein Unterprogramm bzw. eine Kommentarzeile.

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

Mit dem Batchbefehl goto wird die Sprungmarke angesprungen.

#### Anmerkung:

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

#### Syntax

:NAMEDERSPRUNGMARKE

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

#### Inhalt:

```

1  if exist C:\blabla.txt goto EDITBLA
2  goto END
3
4  :: Kommentarzeile, sofern es erforderlich ist, einen Kommentar zu schreiben
5  :: Zur Unterscheidung von Sprungmarken verwende ich zwei "::"
6  :EDITBLA
7  edit c:\blabla.txt
8
9  :END
  
```

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

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

### 4.1.3 CALL

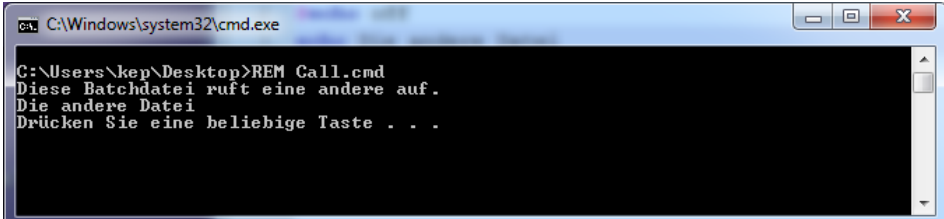
Mit **call** kann man eine andere Batch-Datei aufrufen. Sobald diese beendet wurde, wird die ursprüngliche Batchdatei weiter ausgeführt.

**Beispiel:**

```
1 REM Call.cmd
2 @echo off
3 echo Diese Batchdatei ruft eine andere auf.
4 call anderedatei.bat
5 pause
```

```
1 REM anderedatei.bat
2 @echo off
3 echo Die andere Datei
```

**Die Ausgabe, wenn man die Datei Call.cmd startet:**



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

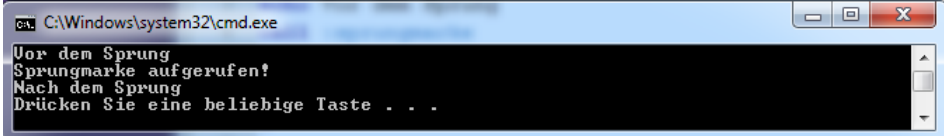
### Beispiel:

```

1  @echo off
2  REM Diese Batchdatei ruft eine eigene Sprungmarke auf
3  echo Vor dem Sprung
4  call :sprungmarke
5  echo Nach dem Sprung
6  pause
7  goto end
8
9  :sprungmarke
10 echo Sprungmarke aufgerufen!
11 goto :eof
12 :: ":EOF" führt nicht zum unsichtbaren Ende der Batch-Datei,
13    wie oben beschrieben,
14    sondern führt die Batch-Datei nach dem Aufruf der
15    Sprungmarke fort
16
17 :end
18 exit

```

### Ausgabe:



### Beispiel mit Parameter:

```

1  @echo off
2  REM Diese Batchdatei ruft eine eigene Sprungmarke auf
3  echo Vor dem Sprung
4  call :sprungmarke meinParameter
5  echo Nach dem Sprung
6  pause
7  goto end
8
9  :sprungmarke
10 echo Sprungmarke aufgerufen und Parameter %1 uebergeben!
11 goto :eof
12
13 :end
14 exit

```

**Ausgabe:**

```
goto :eof
```

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

#### 4.1.4 CLS

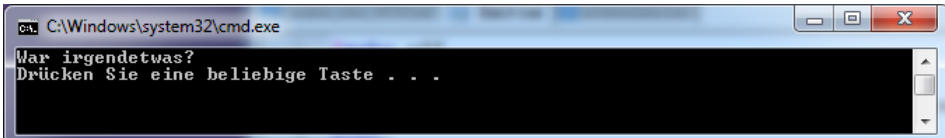
Mit **cls** (clear screen) wird der Bildschirm gelöscht.

**Syntax:**

```
cls
```

**Beispiel:**

```
1 @echo off
2 echo Hier schreibe ich jetzt ganz viel Text.
3 echo Hier kann ich z.B. hinschreiben, dass ich jemanden mag.
4 echo.
5 echo Aber den Text sieht man gleich sowieso nicht mehr... Hihi!
6 cls
7 echo War irgendetwas?
8 pause
```

**Ausgabe:**

### 4.1.5 COLOR

Mit dem Befehl COLOR kann man die Vorder- und Hintergrundfarbe verändern. Die COLOR Werte bestehen aus zwei HEX-Werten. Jede Ziffer kann einen der folgenden Werte haben:

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

#### Beispiel:

```
1 COLOR FC
2
3 pause
```

ergibt z. B. einen weißen Hintergrund mit roter Schrift.



### 4.1.6 ECHO

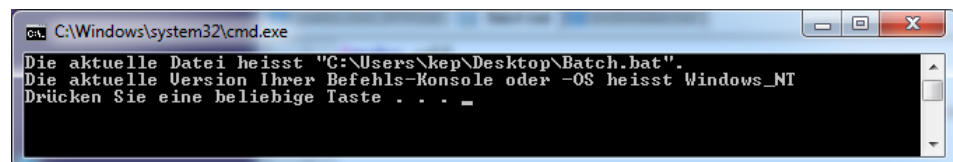
Gibt einen Text aus oder schaltet die Befehlszeilen an/aus. Wenn ein Text ausgegeben wird, können dort auch Variablen angezeigt werden, wie z. B. die Variable %os%

**Syntax:**

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

**Beispiel:**

```
1 @echo off
2 echo Die aktuelle Datei heisst %0.
3 echo Die aktuelle Version Ihrer Befehls-Konsole oder -OS
  heisst %os%
4
5 pause
```

**Beispiel leere Zeilen:**

```
1 @echo off
2 echo Jetzt gibt es 3 Leere Zeilen zu sehen!
3 echo.
4 echo.
5 echo.
6 echo So! Da waren sie.
7
8 pause
```

## 4.1.7 FOR

Ermöglicht die Schleifenbearbeitung.

**Syntax:**

for Variable in Satz do Befehl [Parameter]

**Beispiel:**

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



## ACHTUNG:

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

## FALSCH

*for Variable in Satz do(*

## RICHTIG

*for Variable in Satz do (*

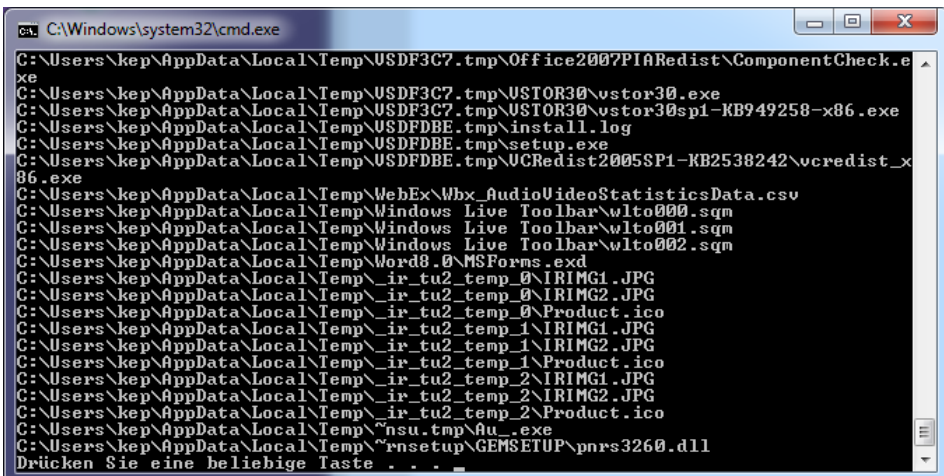
*Befehl1*

*Befehl2*

*)*

```
1 @echo off
2 for /R %temp% %%f in (*.*) do (
3     echo %%f
4 )
5 REM Den Befehl könnte man auch einzeilig schreiben.
6 pause
```

## Ausgabe:



```
C:\Windows\system32\cmd.exe
C:\Users\kep\AppData\Local\Temp\USDF3C7.tmp\Office2007PIARedist\ComponentCheck.exe
C:\Users\kep\AppData\Local\Temp\USDF3C7.tmp\USTOR30\ustor30.exe
C:\Users\kep\AppData\Local\Temp\USDF3C7.tmp\USTOR30\ustor30sp1-KB949258-x86.exe
C:\Users\kep\AppData\Local\Temp\USDFDBE.tmp\install.log
C:\Users\kep\AppData\Local\Temp\USDFDBE.tmp\setup.exe
C:\Users\kep\AppData\Local\Temp\USDFDBE.tmp\UCRedist2005SP1-KB2538242\vc_redist_x86.exe
C:\Users\kep\AppData\Local\Temp\WebEx\Wbx_AudioVideoStatisticsData.csv
C:\Users\kep\AppData\Local\Temp\Windows Live Toolbar\wlto000.sqm
C:\Users\kep\AppData\Local\Temp\Windows Live Toolbar\wlto001.sqm
C:\Users\kep\AppData\Local\Temp\Windows Live Toolbar\wlto002.sqm
C:\Users\kep\AppData\Local\Temp\Word8.0\MSForms.exd
C:\Users\kep\AppData\Local\Temp\ir_tu2_temp_0\IRIMG1.JPG
C:\Users\kep\AppData\Local\Temp\ir_tu2_temp_0\IRIMG2.JPG
C:\Users\kep\AppData\Local\Temp\ir_tu2_temp_0\Product.ico
C:\Users\kep\AppData\Local\Temp\ir_tu2_temp_1\IRIMG1.JPG
C:\Users\kep\AppData\Local\Temp\ir_tu2_temp_1\IRIMG2.JPG
C:\Users\kep\AppData\Local\Temp\ir_tu2_temp_1\Product.ico
C:\Users\kep\AppData\Local\Temp\ir_tu2_temp_2\IRIMG1.JPG
C:\Users\kep\AppData\Local\Temp\ir_tu2_temp_2\IRIMG2.JPG
C:\Users\kep\AppData\Local\Temp\ir_tu2_temp_2\Product.ico
C:\Users\kep\AppData\Local\Temp\rsu.tmp\Au_.exe
C:\Users\kep\AppData\Local\Temp\rsu.tmp\GEMSETUP\pnrs3260.dll
Drücken Sie eine beliebige Taste . . .
```

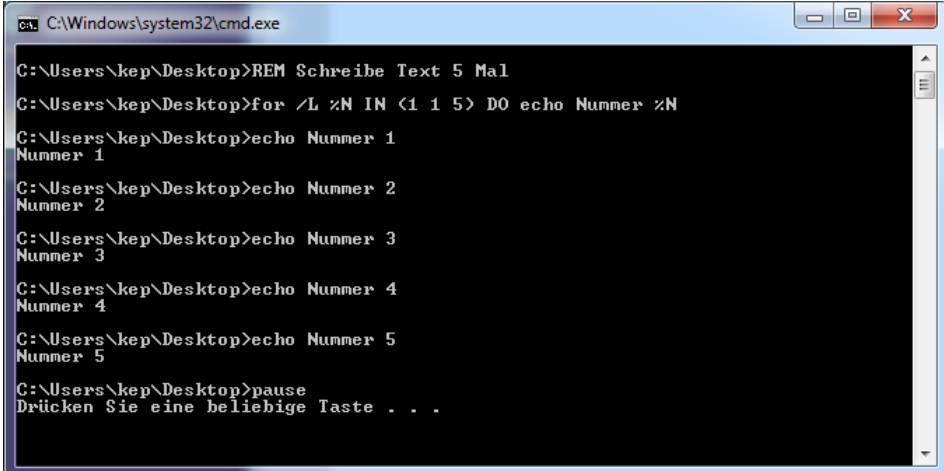
--> Alle Temp-Dateien.

**Zählschleifen:**

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

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

```
1 REM Schreibe Text 5 Mal
2 for /L %%N IN (1, 1, 5) DO echo Nummer %%N
3
4 pause
```

**Ausgabe:**

```
C:\Windows\system32\cmd.exe
C:\Users\kep\Desktop>REM Schreibe Text 5 Mal
C:\Users\kep\Desktop>for /L %N IN (1 1 5) DO echo Nummer %N
C:\Users\kep\Desktop>echo Nummer 1
Nummer 1
C:\Users\kep\Desktop>echo Nummer 2
Nummer 2
C:\Users\kep\Desktop>echo Nummer 3
Nummer 3
C:\Users\kep\Desktop>echo Nummer 4
Nummer 4
C:\Users\kep\Desktop>echo Nummer 5
Nummer 5
C:\Users\kep\Desktop>pause
Drücken Sie eine beliebige Taste . . .
```

### 4.1.8 GOTO

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

**Syntax:**

goto NAMEDERSPRUNGMARKE

**Beispiel:**

Siehe unter : [\(Doppelpunkt\)](#).

### 4.1.9 IF

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

#### Beispiel 1:

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

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

#### Beispiel 2:

```
1 @echo off
2 IF "%COMPUTERNAME%"=="KLZ0052" GOTO WAHR
3 REM hier landet man wenn der if-Ausdruck falsch ist
4 GOTO WEITER
5 :WAHR
6 REM hier landet man wenn der if-Ausdruck wahr ist
7 echo Willkommen Zuhause
8 REM Jetzt wird der if Zweig verlassen
9 GOTO WEITER
10
11 :WEITER
12 echo.Have a nice Day!
13
14 pause
```



#### Beispiel 3:

```
1 @echo off
2 IF "%COMPUTERNAME%"=="KLZ0051" (
3     echo Willkommen zu Hause!
4 ) ELSE (
5     echo Du bist auf Computer: %COMPUTERNAME%
6 )
7 echo.Schoenen Tag noch!
8
9 pause
```

```

C:\Windows\system32\cmd.exe
Du bist auf Computer: KLZ0052
Schoenen Tag noch!
Drücken Sie eine beliebige Taste . . . _

```

### Syntax Vergleiche:

**IF <NOT> Variable1==Variable2**

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

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

**==** ist gleich

**EQU** ist gleich

**NEQ** nicht gleich

**LSS** kleiner als

**LEQ** kleiner als oder gleich

**GTR** größer als

**GEQ** größer als oder gleich

### Hinweis zu UND bzw. ODER Verknüpfung:

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

```

1 set A=true
2 set B=false
3 if "%A%;%B%"=="true,true" (
4     echo A und B sind beide TRUE
5 ) else (
6     echo entweder A oder B sind nicht = TRUE
7 )

```

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

```

1 set A=true
2 set B=false
3 if "%A%"=="true" goto :WAHR // Diese Zeile ist doch erfüllt,
  also sollte der in :WAHR springen
4 if "%B%"=="true" goto :WAHR
5 REM keine der Bedingungen ist zu :WAHR gesprungen, wir sind
  also im ELSE Zweig
6 REM hier waere :FALSCH
7
8 echo Weder A noch B ist TRUE
9 goto :eof
10
11 :WAHR
12 echo A oder B ist TRUE

```

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

```

1 set A=true
2 set B=false
3 if "%A%"NEQ"true" goto :FALSCH
4 if "%B%"NEQ"true" goto :FALSCH
5
6 REM wird sind durch die IFs gekommen, also hat keine der
  Bedingungen angeschlagen.
7 REM hier waere die :WAHR Sprungmarke
8
9 echo A und B sind beide TRUE
10 goto :eof
11
12 :FALSCH
13 echo A oder B (oder beide) sind FALSE

```

Diese beiden Beispiele lassen sich einfach durch Kopieren der "IF..." Zeile um beliebig viele Bedingungen erweitern. Ein mischen von UND und ODER Verknüpfungen ist leider nicht ohne weiteres möglich.

### 4.1.10 PAUSE

Unterbricht die Abarbeitung der Batchdatei und wartet auf einen Tastendruck.

#### Syntax:

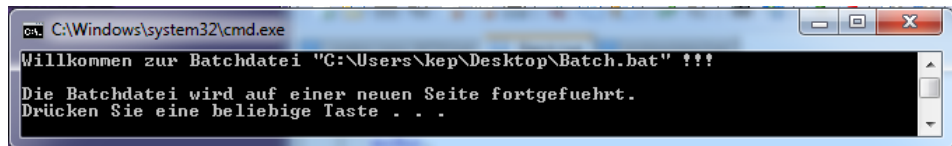
pause

#### Beispiel 1:

```

1 @echo off
2 echo Willkommen zur Batchdatei %0 !!!
3 echo.
4 echo Die Batchdatei wird auf einer neuen Seite fortgefuehrt.
5 pause
6 cls
7 echo Hier faengt meine Batchdatei an...
8 pause

```

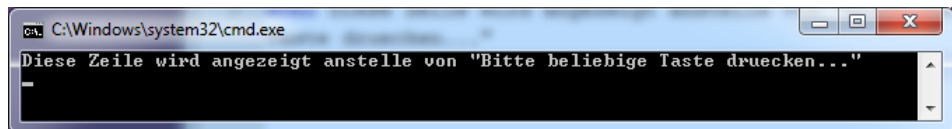


**Beispiel 2:**

```

1 @echo off
2 echo Diese Zeile wird angezeigt anstelle von "Bitte beliebige
  Taste druecken..."
3 pause > NUL

```

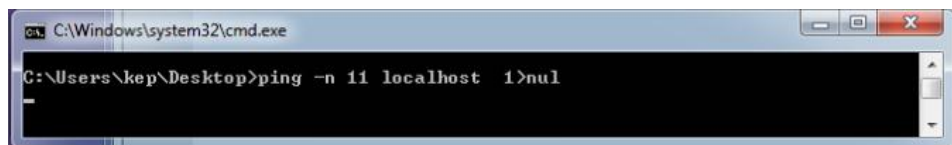


Um eine Pause für eine bestimmte Zeitdauer vorzugeben, kann der ping-Befehl „missbraucht“ werden. Das folgende Beispiel erzeugt eine Pause von etwa 10 Sekunden (nämlich 11 minus 1); durch die Ausgabeumleitung >nul wird jegliche Meldung unterdrückt.

```

1 ping -n 11 localhost >nul

```



### 4.1.11 PUSHD / POPD

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

popd wechselt zum gespeicherten Pfad.

Die Befehle können geschachtelt werden.

**Syntax:**

pushd pfad

popd

**Beispiel:**

```
1 C:\WINDOWS>pushd c:\temp
2 C:\temp>pushd c:\
3 C:\>popd
4 C:\temp>popd
5 C:\WINDOWS>
```

### 4.1.12 REM

REM leitet einen Kommentar ein. Die Zeile wird ignoriert, beachten Sie jedoch, dass REM von einem Leerzeichen / Tabulator gefolgt werden muss. Alternativ dazu werden häufig auch Sprungmarken eingesetzt, da hier nur ein statt vier Zeichen verwendet werden muss. Häufig wird jedoch die Sprungmarke zur besseren Übersicht doppelt hintereinander geschrieben ::

**Beispiel:**

```
1 REM kill iexplore.exe
2 :kill iexplore.exe
3 ::kill iexplore.exe
```

### 4.1.13 START

Startet ein Programm.

**Syntax:**

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

Optionen:

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

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

regedit.exe = neuer Registrierungseditor

(regedt32.exe = älterer Registrierungseditor)

explorer.exe = Windows Ordner Explorer

taskmgr.exe = Windows Taskmanager

taskeng.exe = Aufgabenplanungsmodul

calc.exe = Taschenrechner

mshta.exe = Scripthost für HTA (HTML) Scripting

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

iexplore.exe = Microsoft Windows Internet Explorer, Standardbrowser für Windows

firefox.exe = Mozilla Firefox, Internetbrowser

dialer.exe = Windows-Hilfeprogramm für DFÜ Einwahlverbindungen

Notepad.exe = Standard-Textbearbeitungsprogramm von Microsoft

cmd.exe = Microsoft Windows Befehlsprozessor für Batch

winword = Microsoft Word



## 4.1.14 Variablen

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

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

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

Nützliche Stringoperation für Variablen:

**Teilstring:**

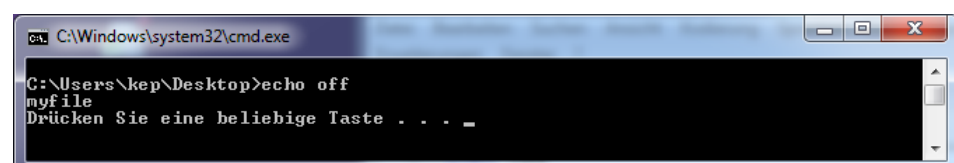
**Syntax:**

`var:~n,m`

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

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

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



**Stringsstitution:****Syntax:**`var:str1=str2`

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

```
1 echo off
2 set str="mycommand /p /m file"
3 echo %str%
4 set str=%str:/p /m=/t %
5 echo %str%
6
7 pause
```

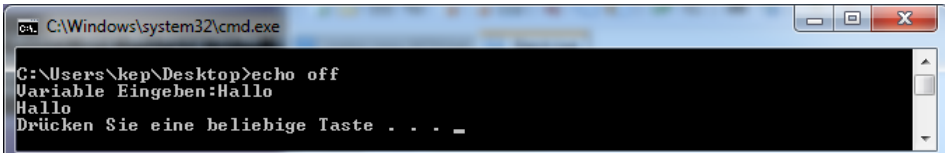


```
C:\Windows\system32\cmd.exe
C:\Users\kep\Desktop>echo off
"mycommand /p /m file"
"mycommand /t file"
Drücken Sie eine beliebige Taste . . . _
```

**Benutzereingaben in Variablen speichern:**

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

```
1 echo off
2 set /p EINGABE=Variable Eingeben:
3 echo %EINGABE%
4
5 pause
```



```
C:\Windows\system32\cmd.exe
C:\Users\kep\Desktop>echo off
Variable Eingeben:Hallo
Hallo
Drücken Sie eine beliebige Taste . . . _
```



