Grundlagen der Windows-Konsole

Die **Windows-Konsole**, **Kommandozeile** oder **Befehlszeile** ist eine Anwendung welche Textzeilen als Eingabe vom Benutzer entgegennimmt und Befehle ausführt. Im Englischen wird sie als **command-line** oder *command prompt* (Eingabeaufforderung) bezeichnet.

Eine Anwendung unter Windows kann entweder eine grafische Benutzeroberfläche (*Graphical User Interface*, GUI) haben oder als **Konsolenanwendung** umgesetzt sein. Solche rein über textbasierte Kommandos gesteuerten Konsolenanwendungen werden in der Windows-Konsole ausgeführt. In einer ausführbaren Datei (mit der Endung **.exe**) wird bei der Erstellung festgelegt ob es sich um eine reine Windows-GUI oder eine Konsolenanwendung handelt.

Eingabeaufforderung

Um die **Windows-Konsole** zu starten muss aus dem Start-Menü die **Eingabeaufforderung** (`C:\Windows\System32\cmd.exe`) ausgewählt werden.

Tastenkürzel

Tastenkürzel	Beschreibung
ESC	Löscht die aktuelle Kommandozeile
F7	Zeigt Befehlshistorie an
F11	Vollbild
Strg + C	Bricht den aktuelle laufen den Befehl bzw. das Programm ab
Strg + Z	Sendet das EOF-Zeichen
TAB	Vervollständigt Datei- und Ordnernamen
↑ / ↓	Scrollt durch die Befehlshistorie

Befehle

Befehl	Beschreibung	Beispiel
`cd`	Wechselt das Verzeichnis aus dem aktuellen Laufwerk (z.B. `E:` um auch das Laufwerk zu wechseln)	`cd \Windows\System32`
`call`	Ruft eine Batchdatei oder Subroutine auf	`call datei.bat`
`cls`	Löscht den Bildschirminhalt	`cls`
`copy`	Kopiert eine oder mehrere Dateien	`copy quelle ziel`
`del`	Löscht eine oder mehrere Dateien	`del *.obj`
`dir`	Listet Dateien und Unterverzeichnisse auf	`dir *.txt`
`echo`	Gibt Meldungen auf Konsole aus	`echo Meldung`
`exit`	Beendet die Konsole oder die aktuelle Batchdatei	`exit 1`
`findstr`	Sucht nach Zeichenfolgen in Dateien	`findstr wort datei.txt`
`goto`	Wechsel zu einer Sprungmarke in einer Batchdatei	`goto next`
`if`	Verarbeitet Ausdrücke in einer Batchdatei abhängig von Bedingungen	`if exist datei.txt echo Gefunden`
`mkdir`	Erstellt ein Verzeichnis	`mkdir ordner1`
`more`	Zeigt Textdateien seitenweise auf dem Bildschirm an	`more datei.txt`
`move`	Verschiebt Dateien und Verzeichnisse	`move a.txt b.txt`
`pause`	Wartet auf einen Tastendruck	`pause`
`rmdir`	Löscht ein Verzeichnis	`rmdir ordner1`
`rem`	Leitet Kommentare (in einer Batchdatei) ein	`rem Ein Kommentar`
`set`	Setzt oder löscht Umgebungsvariablen	`set VAR=Wert`
`type`	Zeigt den Inhalt einer Textdatei an	`type datei.txt`
`where`	Zeigt den Speicherort von Dateien an	`where java`

Befehlszeilenargumente werden den Befehlen mit Leerzeichen getrennt übergeben. Enthält ein einzelnes Argument ein Leerzeichen oder andere Befehlszeichen muss das Argument in Anführungszeichen (") geschrieben werden. In der Regel entspricht eine Zeile einem Befehl. Wenn mehrere Befehle in einer Zeile ausgeführt werden sollen, müssen diese mit dem &-Zeichen getrennt werden.

```
mkdir "Neuer Ordner"
echo Eins & echo Zwei & echo Drei
echo "Lach- & Sachgeschichten"
```

Return Codes

Befehle und Programme geben nach der Ausführung einen Zahlenwert an das aufrufende Programm zurück. Dieser **Return Code** (Rückgabewert; auch **Exit-Status** genannt) kann ausgewertet werden um festzustellen ob das Programm bzw. der Befehl erfolgreich ausgeführt wurde. Jeder Wert ungleich **0** ist ein Fehler-Code. Mit der Umgebungsvariable `%ERRORLEVEL%` lässt sich der Return code des zuletzt ausgeführten Befehls bzw. Programms in der Windows-Konsole anzeigen. Außerdem lassen sich mit `&&` zwei Befehle verknüpfen, wobei der zweite Befehl nur ausgeführt wird, wenn der erste erfolgreich ausgeführt wurde. Mit `||` lassen sich zwei Befehle verknüpfen, wobei der zweite Befehl nur im Fehlerfall des ersten Befehls ausgeführt wird.

```
type NichtVorhanden
echo %ERRORLEVEL%
mkdir "Neuer Ordner" && echo Ordner angelegt
type NichtVorhanden || echo Nicht gefunden
```

Standard Ein- und Ausgabekanäle

Konsolenanwendungen verfügen über einen Kanal für die *Standardeingabe* (0 – **stdin**) sowie jeweils über einen Kanal für die *Standardausgabe* (1 – **stdout**) und die *Standardfehlerausgabe* (2 – **stderr**). Mit den Operatoren `>`, `>>` (*append*) und `<` lassen sich Ausgaben in eine Datei umleiten bzw. Eingaben aus einer Datei einlesen. Mit `|` (Pipe) lassen sich zwei Befehle verknüpfen und die Ausgabe des ersten Befehls als Eingabe des zweiten Befehls verwenden.

```
echo Eins > text.txt
echo Zwei >> text.txt
more < text.txt
type text.txt | findstr /n /i zwei
```

Die Ausgabekanäle lassen sich über ihre Nummer einzeln umleiten bzw. mit `2>&1` am Ende des Befehls zusammenlegen.

```
dir NichtVorhanden 1> out.log 2> err.log
dir NichtVorhanden > ausgabe.log 2>&1
```

Umgebungsvariablen

Als Umgebungsvariable bezeichnet man konfigurierbare Variablen in Betriebssystemen, die oft Pfade zu bestimmten Programmen oder Daten enthalten, sowie bestimmte Daten und Einstellungen, die von mehreren Programmen verwendet werden können. Bei den Werten handelt es sich immer um Zeichenketten.

Die Windows-Konsole stellt von sich aus einige dynamisch generierte Umgebungsvariablen zur Verfügung. Diese werden nicht fest gespeichert, und der Wert wird kurz vor der Ausgabe ermittelt. Beispiele solcher dynamischer Umgebungsvariablen sind **DATE**, **TIME** und **ERRORLEVEL**.

Quelle: Wikipedia (https://de.wikipedia.org/wiki/Umgebungsvariable)

Variable	Beschreibung
`%COMPUTERNAME%`	Rechnername
`%COMSPEC%`	Pfad zum Kommandozeilen-Interpreter (`C:\Windows\System32\cmd.exe`)
`%DATE%`	Aktuelles Datum
`%ERRORLEVEL%`	Fehlercode des zuletzt ausgeführten Befehls bzw. Programms
`%NUMBER_OF_PROCESSORS%`	Anzahl (logischer) Prozessoren bzw. Prozessorkerne des Rechners
`%PATH%`	Suchpfad für Programme und Programmbibliotheken
`%PATHEXT%`	Erweiterungen von ausführbaren Dateien
`%PROGRAMFILES%`	Installationsverzeichnis für Programme
`%TEMP%`	Verzeichnis zur Speicherung von temporären Dateien
`%TIME%`	Aktuelle Uhrzeit
`%SYSTEMROOT%`	System-Verzeichnis (`C:\Windows`)
`%USERNAME%`	Name des aktuellen Benutzers
`%USERPROFILE%`	Pfad zum Benutzerprofil des aktuellen Benutzers
`%0`, `%1`, `%2`, `%3`, `%4`, `%5`, `%6`, `%7`, `%8`, `%9`	Befehlszeilenargumente in einer Batchdatei oder Subroutine

Kommandozeilenparameter

Als **Kommandozeilenparameter** werden zusätzliche Parameter zum Kommando in einer Kommandozeile bezeichnet. Sie werden durch ein Leerzeichen vom Kommando und voneinander getrennt und folgen immer auf das Kommando. Sie gliedern sich in Optionen und Argumente.

Optionen werden auf Unix-artigen Systemen mit einem oder zwei Bindestrichen (`-`) eingeleitet, in der Windows-Welt mit einem Schrägstrich (`/`). Sie modifizieren die Wirkung eines Kommandos. Die meisten Kommandos bieten einen einheitlichen Parameter, der alle für das jeweilige Kommando möglichen Optionen anzeigt. Dies ist unter Linux `--help`, unter Windows ist es `/?`. Mehrere gleichzeitige Optionen dürfen bei manchen Kommandos zusammengefasst werden, beispielsweise kann der Linux-Befehl `ls -l -h` auch als `ls -lh` angegeben werden.

Argumente sind Namen von Dateien, Verzeichnissen, Benutzern oder ähnlichen Objekten, auf die das Kommando angewendet werden soll.

Viele Kommandozeilenanwendung die in Windows verwendet werden wurden von Unix-artigen Systemen portiert oder sind generell auf mehreren Betriebssystemen verfügbar, weswegen Optionen mit Bindestrichen auch hier häufig anzutreffen sind.

Batchdateien

Eine **Batchdatei** (Stapelverarbeitungsdatei) ist eine Textdatei mit der Endung `.bat` oder `.cmd` welche mehrere Befehle aus der Windows-Konsole enthält und mit dem Kommandozeileninterpreter (`cmd.exe`) als Skript ausgeführt werden kann.

Mit `@` am Anfang der Zeile lässt sich die Befehlsausgabe für einen bestimmten Befehl ausschalten. Das Kommando "`@echo off`" schaltet die Befehlsausgabe für das gesamte Skript ab. Außerdem lassen sich mit Klammern mehrere Befehle, z.B. für die Verwendung mit `if`, gruppieren.

```
@rem Befehlsanzeige ausschalten
@echo off

rem Skript beenden, wenn erforderliches Argument fehlt
if "%1" == "" (echo Argument fehlt. 1>&2 & exit /b 1)

if not exist %1 (
    echo %DATE% %TIME% > %1
) else (
    echo Datei bereits vorhanden. 1>&2
    exit /b 2
)
```

Sprungmarken

Innerhalb einer Batchdatei können in Zeilen mit dem `:`-Zeichen am Anfang **Sprungmarken** gesetzt werden. Innerhalb der Batchdatei kann dann mit dem Befehlen `goto` oder `call` zu dieser Zeile gesprungen und die Ausführung dort fortgesetzt werden. Bei der Verwendung von `call` wird die Batchdatei ein zweites Mal, an der Stelle der Sprungmarke aufgerufen und anschließen die Ausführung an der ursprünglichen fortgeführt. Dadurch lassen sich *Subroutinen* ähnlich wie Funktionen realisieren. In Batchdateien mit der Endung `.cmd` gibt es eine *virtuelle* Sprungmarke `:eof` welche das Ende der Datei repräsentiert.

```
@echo off

if /i "%1" == "clean" goto clean

if exist *.pdf goto :eof

echo Creating PDFs...
call :makepdf handout
call :makepdf example

goto :eof

rem Subroutine die mit CALL aufgerufen wird
:makepdf
echo %1.pdf
call scripts\html2pdf.cmd %1.html %1.pdf
goto :eof

rem wird nur mit dem Argument "clean" aufgerufen
:clean
del *.pdf
```