

Administratoren haben häufig die Anforderung, routinemäßige oder umfangreiche Aufgaben zu automatisieren. Windows NT stellt zu diesem Zweck einen Interpreter auf Kommandozeilenebene zur Verfügung, mit dem sich einfache Skripten erstellen lassen.

DIRK PELZER

Windows NT läßt sich mit Hilfe von zahlreichen Kommandos direkt über die MS-DOS-Eingabeaufforderung oder über Batch-, beziehungsweise Skriptprogramme administrieren. Die Ausführung übernimmt dabei ein Kommando-Interpreter, der in der Lage ist, ASCII-Dateien mit der Endung .BAT oder .CMD zeilenweise abzuarbeiten und die enthaltenen Befehle sequentiell auszuführen. Im folgenden werden zunächst grundlegende Befehle und Konzepte erläutert. Weitere Teile dieser Serie behandeln die Themen Benutzer- und Rechteverwaltung, Netzwerkadministration, Prozeßverwaltung, Registry-Manipulation und Troubleshooting.

Übersicht über verfügbare Befehle

Um sich einen Überblick über die verschiedenen Skriptbefehle zu verschaffen, zieht man am besten die Windows NT-Hilfe zu Rate, die eine Liste aller verfügbaren Kommandos beinhaltet. Dazu klickt der Administrator auf den Start-Knopf in der Menüleiste und wählt dann unter „Hilfe“ den Eintrag „Windows NT Befehle“. Wer bereits mit Kommandosprache vertraut ist, kann auch innerhalb der MS-DOS-Eingabeaufforderung den Befehl HELP gefolgt vom Befehlsnamen verwenden, um eine Erklärung zu einem spezifischen Kommando zu erhalten.

Aufbau eines Batch-Programms

Batch-Programme unter Windows-NT haben im Gegensatz zu Programmen höherer Programmiersprachen wie C keine besonderen Regeln für den Aufbau. So müssen weder Variable noch Funktionen deklariert werden. Ein Skript wird im einfachsten Fall Zeile für Zeile abgearbeitet. Es empfiehlt sich jedoch, wie in jeder anderen Programmiersprache auch, auf eine gewisse Struktur zu achten und Aktionen möglichst sinnvoll zu dokumentieren.

Sprungbefehl in Batch-Programmen

Um ein Batch-Programm nicht nur Zeile für Zeile abarbeiten zu können, sondern um auch absolute oder bedingte Sprünge ausführen zu können, existiert der Sprungbefehl GOTO, ähnlich wie in der Programmiersprache BASIC. Als Sprungziel wird ein sogenanntes Label verwendet. Dieses wird mit einem vorangestellten Doppelpunkt gekennzeichnet. Um die Lesbarkeit eines Batch-Programms nicht unnötig zu erschweren, sollte man den GOTO-Befehl allerdings sparsam einsetzen, da ansonsten unschöner „Spaghetti-Code“ entsteht, bei dem die Befehle kreuz und quer ausgeführt werden.

Beispiel:

```
.  
. .  
. .  
. REM Befehle vor GOTO  
. .  
. GOTO Label1  
. .  
. .  
. :Label1  
. REM Befehle, die nach dem Sprungziel Label1  
. ausgeführt werden  
. .  
. .
```

In dem Beispiel werden zunächst der Reihe nach alle Befehle abgearbeitet, die vor dem GOTO-Befehl stehen. Sobald das Kommando GOTO Label1 erreicht wird, wird die Abarbeitung in der Zeile fortgesetzt, die auf das mit :Label1 markierte Sprungziel folgt.

Kommentare

Um ein Batch-Programm lesbarer zu gestalten, empfiehlt sich der Einsatz von Kommentaren, die mit dem Schlüsselwort REM für Remark eingeleitet werden und mit dem Zeilenvorschub enden.

Beispiel:

REM Dies ist eine Kommentarzeile und wird ignoriert.

Textausgabe in Batch-Programmen

Häufig ergibt sich die Notwendigkeit, bei der Ausführung eines Skripts Texte auf dem Bildschirm auszugeben, um beispielsweise den Anwender über Erfolg oder Mißerfolg einer Aktion zu informieren. Hierfür wird das Kommando ECHO verwendet.

Beispiel:

ECHO Das Skript wurde erfolgreich abgearbeitet.

ECHO hat aber noch eine andere Funktion. Bei der Abarbeitung eines Batch-Programms wird zunächst einmal jede Zeile auf dem Bildschirm ausgegeben, bevor das entsprechende Kommando ausgeführt wird. Um das zu vermeiden fügt man in die Batch-Datei das Kommando ECHO OFF ein. Dieses sorgt dafür, daß die im Skript stehenden Befehle vor der Ausführung nicht mehr auf dem Bildschirm ausgegeben werden, sondern nur noch die Texte, Fehlermeldungen und Kommentare, die die Kommandos selbst zurückliefern. Um die Bildschirmausgabe wieder zu aktivieren, verwendet man ECHO ON.

Parameterübergabe an Batchprogramme

Ähnlich wie bei Kommandozeilen-Programmen, kann es auch bei Skripten notwendig sein, vor der Ausführung Parameter zu übergeben, die während der Laufzeit ausgewertet werden und den Ablauf beeinflussen. Prinzipiell können einer Batch-Datei bis zu neun Parameter übergeben werden, die über die Bezeichner %1, %2, %3 bis %9 zur Verfügung stehen. Über %0 kann man den Namen der ausgeführten Batch-Datei erhalten. Es ist zu beachten, daß bei der Parameterübergabe zwischen Groß- und Kleinschreibung unterschieden wird. Der Aufruf von SKRIPT.BAT/a und von SKRIPT.BAT /A hat also prinzipiell eine unterschiedliche Bedeutung, die innerhalb des Skriptes berücksichtigt werden muß.

Beispiel:

*@ECHO OFF
REM Skript zur Demonstration der Parameter-*

übergabe

IF "%1" == "" GOTO Fehler

IF "%1" == "/F" GOTO ParameterF

IF "%1" == "/f" GOTO ParameterF

IF "%1" == "/?" GOTO Hilfe

@ECHO Das Skript %0 wurde mit falschem Parameter aufgerufen.

GOTO Ende

:Fehler

@ECHO Fehler: Es wurde kein Parameter übergeben.

GOTO Ende

:ParameterF

@ECHO Dem Skript %0 wurde der Parameter %1 übergeben.

GOTO Ende

:Hilfe

@ECHO Das Skript %0 muß mit mindestens einem Parameter aufgerufen werden.

:Ende

@ECHO ON

Wenn das Skript den Namen TEST.BAT erhält, so führt der Aufruf mit unterschiedlichen Parametern zu folgenden Ergebnissen:

Aufruf	Ergebnis
TEST	Fehler: Es wurde kein Parameter übergeben.
TEST/?	Das Skript TEST muß mit mindestens einem Parameter aufgerufen werden.
TEST/f	Dem Skript TEST wurde der Parameter /f übergeben.
TEST/F	Dem Skript TEST wurde der Parameter /F übergeben.
TEST/g	Das Skript TEST wurde mit falschem Parameter aufgerufen.

Umgebungsvariable

Zusätzlich zu den Parametern %1, %2 etc., die einem Batch-Programm beim Aufruf übergeben werden können, besteht auch die Möglichkeit, Umgebungsvariablen zu verwenden. Bei Benutzung in Skripten, sind die Namen der Umgebungsvariablen in Prozent-Zeichen einzuschließen. Während der Abarbeitung des Batch-Programms durch den Kommando-Interpreter wird die so gekennzeichnete Umgebungsvariable durch deren

ten Umgebungsvariablen zu erhalten, gibt man auf einer Kommandozeile den Befehl SET ein. Dieser liefert alle aktuell definierten Umgebungsvariablen zurück.

Über SET lassen sich aber auch neue Umgebungs-variable definieren oder bestehende mit anderen Werten versehen. Die Syntax dafür lautet:

```
SET <Umgebungsvariable>=<Wert>
```

Beispiel

```
SETOPTICAL_DRIVE=G:\
IF %OPTICAL_DRIVE% == G:\ECHO
Standardlaufwerk gefunden
```

Bedingungen in Batch-Programmen prüfen

Während der Abarbeitung eines Batch-Programmes kann es notwendig sein, in Abhängigkeit von bestimmten Ereignissen, an unterschiedlichen Stellen des Skriptes fortzufahren oder ein bestimmtes Kommando auszuführen. Zu diesem Zweck existiert eine Abfragemöglichkeit mit dem Befehl IF. Ist eine Bedingung erfüllt, so wird der Rest der Zeile hinter der Bedingung abgearbeitet. Andernfalls setzt die Programmausführung mit der nächsten Befehlszeile fort. Die Syntax des IF-Befehls lautet:

```
IF <Bedingung> <Befehl>
```

IF kann dazu verwendet werden, die Existenz von Objekten, die Werte von Variablen, Zeichenfolgen oder Rückgabewerte von Programmen abzufragen. Ein Objekt kann dabei beispielsweise eine Datei, ein Verzeichnis oder ein Gerät sein, wie beispielsweise ein serieller Anschluß.

Beispiel:

```
IF NOT EXIST C:\TEMP MD C:\TEMP
```

In diesem Beispiel wird über den IF-Befehl festgestellt, ob das Verzeichnis C:\TEMP existiert und wenn nicht, dann wird es über das Kommando MD (Make Directory) angelegt. IF wurde in diesem Beispiel mit dem Booleschen Operator NOT verknüpft, was dazu dient, Schreiarbeit zu sparen. Alternativ hätte man auch folgendes Skript schreiben

können:

```
IF EXIST C:\TEMP GOTO Existiert
MD C:\TEMP
:Existiert
```

Um den Wert einer Zeichenfolge abzufragen, verwendet man folgende Syntax:

```
IF <Zeichenfolge1> == <Zeichenfolge2> <Befehl>
```

Beispiel:

```
IF %SystemRoot% == C:\WINNT ECHO NT-
Standardpfad gefunden
```

ERRORLEVEL verwenden

Abhängig davon, ob ein Programm erfolgreich ausgeführt wurde, liefert es normalerweise einen Rückgabewert, der im ERRORLEVEL gespeichert wird. Die gängige Konvention ist, daß ein erfolgreich ausgeführtes Programm einen ERRORLEVEL mit dem Wert 0 zurückliefert. Jeder von 0 verschiedene Wert bedeutet einen Fehler, wobei der ERRORLEVEL dann in der Regel mit einer Fehlernummer des aufgerufenen Programms übereinstimmt. Dies kann so sein, muß aber nicht, denn es gibt keine zwingende Vorgabe, daß ein Programm einen Rückgabewert liefern muß. Mit Hilfe des IF-Befehls und dem ERRORLEVEL können in einem Skript Fehlerbehandlungsroutinen implementiert werden.

Beispiel:

```
@ECHO OFF
CHKDSK C:/F/R
IF ERRORLEVEL 3 GOTO Error3
@ECHO CHKDSK wurde erfolgreich durchgeführt.
GOTO End
:Error3
@ECHO CHKDSK konnte keinen exklusiven
Zugriff auf Laufwerk C: erhalten.
:End
@ECHO ON
```

In dem Beispiel wird versucht, das CHKDSK-Kommando auf Laufwerk C: auszuführen. Wenn NT nicht in der Lage ist, exklusiven Zugriff auf das Laufwerk zu bekommen, erhält man eine Fehlermeldung und als Rückgabewert einen ERRORLEVEL vom Wert 3.

Um herauszufinden, welcher Befehl welchen ERRORLEVEL zurückliefert kann man ein kleines Batch-Programm verwenden, welches den Namen ELEVEL.BAT bekommen soll und folgendermaßen aufgebaut ist:

```
@ECHO OFF
%1 %2 %3 %4 %5 %6 %7 %8 %9
@ECHO Fehlernummer: %ERRORLEVEL%
@ECHO ON
```

Wenn der Administrator nun herausfinden möchte, welche ERRORLEVEL von einem bestimmten Kommando unter bestimmten Bedingungen zurückgeliefert werden, führt er ELEVEL.BAT mit folgender Syntax aus:

```
ELEVEL <Kommando> <Parameter>
```

Um also beispielsweise zu ermitteln, welchen ERRORLEVEL das Kommando RENAME zurückliefert, wenn versucht wird, eine nicht existierende Datei umzubenennen, so führt man folgendes aus:

```
ELEVEL RENAME C:\UVWXYZ.TXT *.OLD
```

Wenn die Datei nicht existiert, erhält man folgendes Ergebnis:

```
Das System kann die angegebene Datei nicht finden.Fehlernummer: 1
```

Es wird also zunächst die Original-Fehlermeldung des RENAME-Kommandos ausgegeben und anschließend der ERRORLEVEL. Diesen kann der Administrator dazu verwenden, eine Fehlerbehandlungsroutine aufzurufen.

Schleifenprogrammierung in Batch-Dateien

Eine überaus nützliche Funktion, die in Batch-Programmen zum Einsatz kommen kann, ist das Kommando FOR. Mit diesem lassen sich Befehle in einer Schleife mehrfach ausführen. Der FOR-Befehl hat teilweise eine sehr komplexe Syntax, die jedoch in der NT-Hilfe ausführlich erläutert wird.

Zur Einführung soll ein kleines Beispiel dienen, mit dessen Hilfe alle komprimierten DLL-Dateien der Windows-NT CD, die die Datei-Endung *.DL_ haben expandiert und in das Verzeichnis C:\TEMP kopiert werden.

Die allgemeine Syntax des FOR-Kommandos lautet folgendermaßen:

```
FOR %%P IN (<Dateiliste>) DO <Kommando>
[Parameter]
```

Dabei ist <Dateiliste> eine Liste der Dateien, für die der unter <Kommando> eingetragene Befehl mit den optionalen Parametern gemäß der unter [Parameter] angegebenen Liste ausgeführt werden soll. In der <Dateiliste> können dabei mehrere durch Leerzeichen separierte Ausdrücke stehen. Ebenso können unter [Parameter] mehrere durch Leerzeichen separierte Parameter stehen. Für das beschriebene Beispiel der Expandierung aller DLL-Dateien kann demnach folgende Syntax zum Einsatz kommen:

```
FOR %%P IN (*.DL_) DO EXPAND %%P
C:\TEMP-R
```

Bei der Ausführung des Kommandos ist folgendes zu beachten:

- Der Befehl muß in dem Verzeichnis ausgeführt werden, in dem sich die mit *.DL_ bezeichneten Dateien befinden. Wenn sich also die NT Installations-CD im Laufwerk F: befindet, sollte man vorher mit dem Befehl CD /D F:\I386 in das entsprechende Verzeichnis wechseln. Der Parameter /D beim Aufruf von CD bewirkt, daß auch automatisch auf das im Pfad angegebenen Laufwerk gewechselt wird.
- Möchte der Administrator die FOR-Schleife nicht nur auf Dateien mit der Endung *.DL_ angewendet, sondern beispielsweise auch auf Dateien mit der Endung *.EX_, so müßte das Kommando folgendermaßen abgewandelt werden: FOR %%P IN (*.DL_ *.EX_) DO EXPAND %%P C:\TEMP-R.
- Wenn man die FOR-Schleife nicht in einer Batch-Datei, sondern direkt von der Kommandozeile ausführen möchte, so darf dem verwendeten Parameter nur ein Prozent-Zeichen vorangestellt werden, also anstelle von %%P ist %P zu verwenden. Andernfalls kommt es zu folgender Fehlermeldung:

```
"%%P" ist syntaktisch an dieser Stelle nicht
verarbeitbar.
```

Synchrones und asynchrones Ausführen von Kommandos und Programmen

Da Windows-NT anders als MS-DOS ein Multitasking-Betriebssystem ist, kann es bei der Ausführung von Kommandos zu Problemen kommen, wenn zwei voneinander abhängige Befehle nacheinander ausgeführt werden und der zweite das Ergebnis des ersten benötigt, um korrekt arbeiten zu können.

Die Ursache dieses Problems ist, daß bestimmte Befehle asynchron ausgeführt werden. Das bedeutet, sie melden dem Kommando-Interpreter, daß sie abgeschlossen sind, obwohl sie noch ausgeführt werden. Der Kommando-Interpreter seinerseits fährt mit der Ausführung des nächsten Befehls fort, der jedoch möglicherweise unsinnige oder falsche Resultate zurück liefert, da er nicht über die Ergebnisse des zuvor ausgeführten Kommandos verfügt.

Um das Problem zu beheben, kann man den Befehl START benutzen, der unter Verwendung entsprechender Parameter dafür sorgt, daß sich ein Programm erst dann zurückmeldet, wenn es tatsächlich beendet ist.

Der Server-Manager von Windows-NT ist ein Beispiel für eine solche Anwendung. Ruft man diesen über den Befehl

SRVMGR.EXE

in einem DOS-Eingabefenster auf, so kehrt die Eingabeaufforderung sofort wieder zurück, obwohl das Programm noch läuft. Führt man hingegen das Kommando:

START /WAIT SRVMGR.EXE

aus, so kehrt die Eingabeaufforderung erst zurück, wenn der Server-Manager wieder beendet ist.

Leider funktioniert das nicht mit allen Programmen, so kehrt beispielsweise die Eingabeaufforderung auch dann sofort zurück, wenn man das NT-Programm Word-Pad mit dem Kommando *START /WAIT WRITE.EXE* ausführt. Es ist demzufolge offensichtlich auch davon abhängig wie eine Anwendung programmiert ist, ob sie sich vom START-Kommando beeinflussen läßt, oder nicht. So bleibt dem Administrator, der START mit der WAIT-Option verwenden möchte nichts anderes übrig, als auszuprobieren, ob die betreffende Anwendung wie erwartet reagiert.

Shortcut

Executive Summary

Über Kommandozeilen-Programme lassen sich zahlreiche administrative Aufgaben, wie zum Beispiel Backups oder die Einrichtung von Benutzern und Shares automatisieren. Windows NT liefert eine ganze Reihe entsprechender Werkzeuge, die für Automatisierungsaufgaben in Skripten verwendet werden können. Diese Serie vermittelt anhand zahlreicher Beispiele die Grundlagen erfolgreicher Skriptprogrammierung unter Windows NT.

Zur Person

DIPL. ING. DIRK PELZER arbeitet als freier Consultant und Journalist in München. Er betreibt ein Storage Labor für verschiedene namhafte Fachzeitschriften. Zudem beschäftigt er sich mit Speichernetzen und Hochverfügbarkeit.