Seminararbeit

99 bottles of beer in DOS-batch

Christoph Zwerschina

christoph.zwerschina@student.uibk.ac.at

23. Februar 2011

Betreuer: Dr. Christian Sternagel

Abstract

In the course of the Specialisation seminar "99 bottles of beer" the programming language DOS-batch will be introduced. Chapter 1 illuminates the history and meaning. In addition, a parallel between DOS-batch and bash in Linux will be drawn. The following chapters will address more specifically programming with DOS-batch.

Inhaltsverzeichnis

1 Geschichte der Programmiersprache DOS-batch

Die geschichtlichen Grundlagen der DOS-batch Programmiersprache wurden aus [1] und [4] entnommen. Die Entstehung von MS-DOS und damit DOS-batch ist eine höchst interessante und einzigartige Geschichte, die mit einem Vertrag zwischen den Microsoft-Bossen Bill Gates und Steve Ballmer auf der einen Seite und IBM auf der anderen Seite beginnt.

1.1 Entstehung

IBM wurde erlaubt PCs mit einem Betriebssystem "DOS" auszustatten, die Rechte der Software blieben aber allein bei Microsoft. Zu dieser Zeit, im Jahre 1981, hatte Microsoft jedoch noch kein "DOS" Betriebssystem. "Q-DOS" wurde von einer kleinen Firma namens Seattle Computers lizensiert und kurze Zeit später für 50.000$ ganz aufgekauft. Der Name Q-DOS (Quick and Dirty Operating System) wurde dabei zu MS-DOS (Microsoft - Disk Operating System).

1.2 Entwicklung

Seit 1981 wurde MS-DOS und damit DOS-batch von Version 1.0 bis zu 7.22 weiterentwickelt. Während dieser Zeit kam zunehmend ein neues System namens "Windows"zum Einsatz. Mit Windows 95 wurde DOS schließlich als Betriebssystem abgelöst und verlor zunehmend an Boden. Trotzdem war in dieser Software ein fixer Bestandteil die "Eingabeaufforderung". Dabei wurde in einem Fenster MS-DOS simuliert, und somit die Batch Programmierung ermöglicht. Auch heute noch ist diese Eingabeaufforderung ein fixer Bestandteil von Windows Betriebssystemen.

1.3 Allgemeine Informationen

Der Name "Batch" kommt aus dem Englischen und bedeutet "Stapel", dies deutet auch die Hauptaufgabe der Sprache an: Sie dient der Abarbeitung von Systemroutinen, die häufig ausgeführt werden. Eine der berühmtesten Batch-Programmdateien stellt die "autoxec.bat" dar. Sie wurde bei jedem Systemstart von MS-DOS Systemen durchgeführt um Aufgaben, wie Einbindung von Peripherie (Drucker, Maus, CD-Rom, Speicherverwaltung, etc.) zu gewährleisten. MS-DOS integriert in seinem Systemkern, der "Command.com" die Batch-Sprache als fixen Bestandteil. Das bedeutet, sobald ein PC mit MS-DOS betrieben wird, ist es auch möglich Batch Programme zu erstellen. Eine Batchprogrammierung kann in jedem ASCII Editor stattfinden, wichtig dabei ist lediglich

die Datei mit der Endung "`bat`" zu versehen. Sobald die Datei so benannt wurde kann sie durch Aufrufen ausgeführt werden. Ein Kompilieren, wie es in vielen Highlevel Sprachen erforderlich ist, ist nicht erforderlich.

1.4 Parallelen zu Linux-bash

Während Microsoft mit MS-DOS und aktuell Windows bis heute einen hohen Marktanteil realisiert, wurde auf "Open Source" Basis eine Betriebssystemgruppe namens "Linux" immer bekannter. Dieses Softwaresystem das Ähnlichkeit mit Unix aufweist, besitzt ein DOS-batch ähnliches Werkzeug mit dem Namen "bash" ("born again shell"). Während Microsoft kaum direkten Zugriff auf interne Hardware erlaubt, kann Linux und damit bash sehr tief in die diesen Bereich eingreifen. Bash ist wesentlich mächtiger und kann mit einer umfangreichen Highlevel Programmiersprache wie C verglichen werden, während DOS-batch nicht diese Funktionalität aufweist. Leider würde ein genauerer Vergleich dieser beiden Sprachen den Umfang dieser Arbeit sprengen, und deshalb wird nur kurz auf Ähnlichkeiten hingewiesen. Trotzdem sind zwei der nachfolgenden Codebeispiele nämlich "Fibonacci" und "Bier" ebenfalls in bash erstellt worden, um Ähnlichkeiten und Programmierung in bash zu zeigen.

2 Programmierung

2.1 Grundregeln

Um ein Programm zu verstehen, bzw. selber erstellen zu können sollten einige Grundregeln beherrscht werden. Diese wurden aus [3] recherchiert. DOS-batch ist nicht geeignet um grafische Anwendungen zu gestalten. Es lassen sich zwar mittels Textsymbolen sehr simple Elemente, beispielsweise ein Rahmen um ein Menü, darstellen, in der Praxis wird dies aber nicht (mehr) verwendet. Viele andere Programmiersprachen wie zum Beispiel Java und C sind für diese Zwecke viel besser geeignet. Die Hauptaufgaben von DOS-batch liegen in der Abarbeitung von immer wiederkehrenden Aufgaben, die zusammengefasst werden sollen - ein zeilenweises ausführen von Befehlen um beispielsweise Dateien nach einem bestimmten Muster zu ordnen, auszufiltern und umzubenennen. In den ersten Versionen von DOS-batch war noch nicht mehr möglich. Mit der Weiterentwicklung und steigenden Versionsnummern wurden immer mehr Möglichkeiten, wie zum Beispiel eine for Schleife, ein If/Else Konstrukt und Arithmetik integriert. Ganz im Gegensatz zu anderen Sprachen wie Linux-bash ist DOS-batch nicht Case sensitive. Wie im zugrundeliegenden DOS können sowohl für Dateinamen als auch für Befehle Gross und Kleinschreibung gemischt werden, ohne dass dies

zu einer Unterscheidung oder einem Fehler führt.
Beim Erstellen einer Programmdatei gilt zu beachten, dass die Zeichenlänge pro Zeile begrenzt ist. Früher lag diese Grenze bei 127 Zeichen. In späteren Windowsversionen wurde diese Restriktion allerdings gelockert - heutzutage können bis zu 255 Zeichen (mit eingesetzten Variablen sogar 1023 Zeichen) pro Zeile verwendet werden. Da in einer solchen Zeile oftmals Variablen stehen, die vor der Abarbeitung ersetzt werden, kann eine solche Länge schnell erreicht werden.
Des Weiteren gibt es nur drei Zahlensystemen, nämlich Oktal, Dezimal und Hexadezimal. Kommazahlen gibt es nicht. Gekennzeichnet werden die unterschiedlichen Zahlensysteme direkt bei deren Erwähnung:

- **Oktal**
  078 (Präfix "0")
- **Dezimal**
  32
- **Hexadezimal**
  0x56 (Präfix "0x")

2.2 DOS-batch Befehle/Operatoren

Die im Folgenden präsentierten Befehle und Operatoren wurden [2] entnommen. Die am meisten benutzten Befehle mit deren Hilfe einfache Programme erstellt werden können lauten:

- `ECHO off`
  Wie schon erwähnt, werden in einer Batchdatei zeilenweise Befehle ausgeführt, diese werden vor Abarbeitung auf die Kommandozeile geschrieben. "ECHO off" schaltet diese Befehlsanzeige aus, nicht aber deren Rückgabe! Mit "ECHO on" wird die Anzeige wieder aktiviert.
- `ECHO <Text>`
  Zeigt nachfolgenden Text auf der Standartausgabe, an.
- `REM <Befehl/Kommentar>`
  Der nachfolgende Befehl wird nicht ausgeführt. Es dient auch um Kommentare im Code zu erzeugen. REM ist gleichbedeutend dem doppelten Doppelpunkt "::", der allerdings erst in späteren Batch-Versionen eingeführt wurde. Es wird oft verwendet um Teile eines Programms vorübergehend nicht auszuführen, ohne sie löschen zu müssen.

- `CALL <Name>`
  Wird benutzt um Dateinamen oder Funktionen aufzurufen. Optional lassen sich Parameter von 1-9 anhängen, die mit `%1` bis `%9` abgefragt werden können.

- `GOTO <Name>`
  Wechselt zu der Sprungmarke "Name". Eine solche Sprungmarke ist durch einen Doppelpunkt vor dem Namen gekennzeichnet.

- `PAUSE`
  Stoppt die Abarbeitung des Programms bis eine Taste gedrückt wird. Es wird eine Meldung ausgegeben: "Drücken Sie eine beliebige Taste..."

- `CLS`
  Löscht den Inhalt des aktuellen DOS Fensters.

- `SET <Name> = <Wert>`
  Mit SET können mehrere Aufgaben erfüllt werden. Grundlegend werden durch SET Variablendeklarationen durchgeführt. Dabei wird in zwei verschiedene Gruppen unterschieden: Umgebungsvariablen und naormale Variablen. Die Umgebungsvariablen werden durch reservierte Bezeichner von normale Variablen unterscheidbar. Wichtige Umgebungsvariablen lauten zum Beispiel:

  - `PROMPT`
    Art der Darstellung der Eingabeaufforderung.
  - `PATH`
    Speichert die Verzeichnispfade, in denen Programme und Befehle gesucht werden.
  - `TEMP`
    Legt den temporären Verzeichnispfad fest.
  - `SYSTEMROOT / WINDIR`
    Zeigt den Ort des Betriebssystems an.

  Durch Eingabe von SET ohne weitere Argumente wird die aktuelle Liste von Umgebungsvariablen angezeigt. Jede Variable lässt sich durch `SET <Name> = ∅` löschen. Wenn eine Variable abgefragt oder verwendet werden soll, wird diese mit anfänglichem und abschließendem "%" geschrieben

  - `echo %PATH%`
  - `echo %userinp%`

Weitere Möglichkeiten, die SET bietet sind:

- `SET /p x=<TEXT>`
  Dies ermöglicht das Einlesen einer Variable x während des Ausführens. Am Bildschirm erscheint der TEXT um dem Benutzer eine Information zu zeigen, welche Eingabe erwartet wird.
- `SET /a <Rechnung>`
  Ermöglicht einfache mathematische Berechnungen. Eine ausführlichere Beschreibung folgt.

Um einzelne Befehle zu verknüpfen stehen folgende Operatoren zur Verfügung:

- `<Befehl1>` & `<Befehl2>`
  Um mehrere Befehle innerhalb einer Zeile ausführen zu können wird "`&`" verwendet. Um diese Arbeitsweise noch verfeinern und nützlicher machen zu können gibt es zwei weitere Operanden, die ähnliche Funktionen bieten.

- `<Befehl1>` && `<Befehl2>`
  für bedingte Befehlsverkettung wird "&&" verwendet. Nur wenn der erste Befehl erfolgreich abgearbeitet wurde, wird auch der zweite Befehl durchgeführt.

- `<Befehl1>` || `<Befehl2>`
  im Gegensatz zu "&&" wird nur beim Fehlschlagen des ersten Befehls der zweite ausgeführt.

Auch arithmetische Operationen sind seit WIN NT4.0 integriert allerdings nur eingeschränkt anwendbar:

```
+, -, *, /, %
```

Addition, Subtraktion, Multiplikation, Division und Modulo stehen zur Verfügung. Wie schon erwähnt lässt sich nur mit ganzen Zahlen rechnen. Die Schreibweise sieht dabei folgendermaßen aus:

```
SET /a x <Operator> y
```

Der Befehl `SET /a` deutet auf eine nachfolgende Rechenoperation hin. Die Variablen x und y können zuvor deklariert werden oder zwei Zahlen aus den drei Zahlensystemen Okt, Hex, Dez. Die Auswertung erfolgt jedoch immer im Dezimalsystem.

2.3 Wichtigste Konstrukte

2.3.1 If / Else

If/Else lässt sich auf zwei unterschiedliche Weisen verwenden. In den ersten Versionen von DOS wurde nur die erste implementiert:

```
IF NOT EXIST <Dateiname> <Befehl>
```

Hier zeigt sich auch wieder eine Hauptaufgabe die ursprünglich mit Dos-batch verbunden wurde, die Dateiorganisation. Das NOT ist optional. Durch Patternmatching wird nach Dateinamen, oder Dateigruppen gesuchen und selektiert. Beim Auftreten (oder nicht auftreten durch NOT)der selektierten Dateigruppen wird der nachfolgende Befehl ausgeführt.
Die zweite Variante der If/Else Schleife ist eine Weiterentwicklung und ähnelt im Umfang den Möglichkeiten die sich auch in Linux bash bieten:

```
IF NOT <Variable1>==<Variable2> <Befehl>
```

Das "==" kann folgend ersetzt werden:

- `EQU` - Ist gleich (äquivalent zu "==", wurde eingeführt, um die Schreibweise zu anderen Vergleichsoperatoren, die nur durch Buchstaben dargestellt werden, anzupassen)
- `NEQ` - Nicht gleich
- `LSS` - Kleiner als
- `LEQ` - Kleiner als oder gleich
- `GTR` - Größer als
- `GEQ` - Größer als oder gleich

```
Bsp.: if %userinp% == 1 goto s1
```

%userinp% Stellt eine Variable dar.

Somit ergibt sich folgende Bedeutung:

"Wenn in der Variable userinp der Wert 1 steht, dann springe nach s1".

2.3.2 for Schleife

Auch hier gibt es zwei Varianten:

```
for %%<Variable> IN <Satz> DO <Befehl>
```

Jedes Element in dem Satz wird als Variable gesehen, und mit dem Befehl ausgeführt.

```
for %%f IN (A B C 1 $) DO echo mitten %%f drin
```

Also lautet die Variable "`f`", der Satz "`(A B C 1 $)`" und der Befehl "`echo mitten %%f drin`". Somit ergibt sich folgende Ausgabe: Eine Erweiterung der

```
C:\>echo mitten A drin
mitten A drin
C:\>echo mitten B drin
mitten B drin
C:\>echo mitten C drin
mitten C drin
C:\>echo mitten 1 drin
mitten 1 drin
C:\>echo mitten $ drin
mitten $ drin
```

for Schleife lässt sich durch einführen des Parameters /L erzeugen:

```
for /L %%N IN (1, 1, 5) DO echo Nummer %%N
```

hier lautet die Variable "`N`", der Satz besteht in diesem Fall immer aus drei Werten (Startwert, Schrittweite, Endwert) "`(1,1,5)`" und der Befehl "`echo Nummer %%N`" Somit ergibt sich folgende Ausgabe:

```
C:\>echo Nummer 1
Nummer 1
C:\>echo Nummer 2
Nummer 2
C:\>echo Nummer 3
Nummer 3
C:\>echo Nummer 4
Nummer 4
C:\>echo Nummer 5
Nummer 5
```

3 Codebeispiele

3.1 Menü

Ziel dieses Beispiels soll die Verwendung einfacher Variablendeklaration mit mathematischer Auswertung, Benutzereingabe und Bildschirmanzeige sein. Alle

```
1  ECHO OFF
2  ECHO Bitte geben Sie 2 Zahlen ein:
3  SET /p user1=bitte 1. Zahl eingeben
4  SET /p user2=bitte 2. Zahl eingeben
5  CLS
6  ECHO Zahl1: %user1%
7  ECHO Zahl2: %user2%
8  ECHO.
9  ECHO 1. Addition
10 ECHO 2. Subtraktion
11 ECHO.
12 SET /p userinp=Bitte Auswahl eingeben (1/2):
13 IF %userinp% == 1 GOTO s1
14 IF %userinp% == 2 GOTO s2
15 GOTO :ende
16 :s1
17 SET /a ergebnis=%user1%+%user2%
18 ECHO.
19 ECHO.
20 ECHO %user1% + %user2% = %ergebnis%
21 GOTO ende
22 :s2
23 SET /a ergebnis = %user1% - %user2%
24 ECHO %user1% - %user2% = %ergebnis%
25 GOTO ende
26 :ende
27 PAUSE
```

Listing 1: Menü

im obigen Beispiel gezeigten Elemente wurden in vorhergehenden Kapiteln erwähnt, sollen hier aber nochmals zur Vertiefung übersichtsmäßig durchgegangen werden:

1. Die Anzeige der einzelnen Befehle wird ausgeschaltet, sodass nur deren Ausgabe oder Rückgabe ersichtlich ist.

2. Zwei Parameter werden vom Benutzer eingelesen, und anschließend mit “1. Addition” oder “2. Subtraktion” auf einem gelöschten Screen ausgegeben.

3. Eine weitere Eingabe des Benutzers wird abgewartet, um an die richtige Stelle mit betreffendem Rechenmodel mittels “goto” zu springen.

4. Mit “pause” wartet das System, um die Ausgabe nicht gleich verschwinden zu lassen, und das Programm zu beenden.

3.2 Fibonacci in DOS-batch

Zusätzlich zu den bisher gezeigten Beispielen wird nachfolgend noch eine Rekursion mit Parameterübergabe konstruiert.

```
1  ECHO OFF
2  SET i=0
3  SET fst=0
4  SET fib=1
5  SET limit=10000
6  CALL:myFibo %fib%,%fst%,%limit%
7  ECHO.Die naechste Fibonacci gleich / groesser %limit%
      ist %NumberN%.
8  PAUSE
9  :myFibo
10
11 SET /a i+=1
12 SET /a Number1=%1
13 SET /a Number2=%2
14 SET /a Limit=%3
15 SET /a NumberN=Number1 + Number2
16
17 ECHO %i%te Fibonacci: %1
18 IF %NumberN% LSS %Limit% CALL:myFibo %NumberN%,%
      Number1%,%Limit%
```

Listing 2: Fibonacci in DOS-batch

1. Zu Beginn werden wieder Variablen deklariert (`i`, `fst`, `fib`, `limit`)

2. Mit `call` wird eine selbsterstellte Funktion ausgeführt, die Parameter (unsere Variablen) akzeptiert.

3. Diese Funktion "`myFibo`" setzt nun neue Variablen: `Number1`,`Number2`,`Limit` und `NumberN`. Die nachfolgende `%1`, `%2`, `%3`, stellt dabei die Parameter des Funktionsaufrufs dar. Der Funktionsaufruf mit eingesetzten Variablen lautete in unserem Fall: "`myfibu 1 0 10000`" daraus folgt

   `Number1 = %1` = 1 berechneter Wert und aktuelle Fibonacci Zahl
   `Number2 = %2` = 0 vorhergehender WeIn the course of the Specialisation seminar "99 bottles of beer" the programming language DOS-batch will be introduced. Chapter 1 illuminates the history and meaning. In addition, a parallel between DOS-batch and bash in Linux will be drawn. The following chapters will address more specifically programming with DOS-batch.rt zur Berechnung

`Limit = %3` = 10000 Limit

`NumberN` zuletzt festgehaltene Fibonacci Zahl
i wird bei jedem erneuten "`call`" inkrementiert um die Anzahl der Rekursionen aufzuzeigen.

4. `If %NumberN% LSS %Limit%` bildet unsere Abbruchbedingung, und stoppt die Rekursion.

5. Mit einem `echo` Befehl wird die Ausgabe im Bereich unseres Limits ausgegeben.

Die Ausgabe sieht wie folgt aus:

```
C:\> echo off
1te Fibonacci: 1
2te Fibonacci: 1
3te Fibonacci: 2
4te Fibonacci: 3
5te Fibonacci: 5
6te Fibonacci: 8
7te Fibonacci: 13
8te Fibonacci: 21
9te Fibonacci: 34
10te Fibonacci: 55
11te Fibonacci: 89
12te Fibonacci: 144
13te Fibonacci: 233
14te Fibonacci: 377
15te Fibonacci: 610
16te Fibonacci: 987
17te Fibonacci: 1597
18te Fibonacci: 2584
19te Fibonacci: 4181
20te Fibonacci: 6765
Die naechste Fibonacci gleich / groesser 10000 ist
   10946.
Druecken Sie eine beliebige Taste...
```

3.3 Fibonacci in bash

In bash lässt sich durch Verwendung einer while Schleife eine einfachere, leichter nachvollziehbare Implementierung erzeugen:

```
1  #!/bin/bash
2  i=1
3  fst=1
4  fib=1
5  limit=10000
6
7  WHILE [ $limit -gt $fib ]
8  DO
9    PRINTF "%dte␣Fibonacci:␣%d\n" $i $fib
10   LET sum=$fib+$fst
11   LET fib=$fst
12   LET fst=$sum
13   LET i=$i+1
14 DONE
```

Listing 3: Fibonacci in bash

1. Auch hier werden Variablen am Anfang deklariert (`i`, `fst`, `fib`, `limit`)
2. Eine while-Schleife wird durch `while` [Bedingung] gestartet.
3. Der Körper der Schleife wird durch `do` eingeleitet und mit einem abschliesendem `done` beendet. In diesem Körper befinden sich Befehle zur Berechnung, und die Ausgabe der aktuell berechneten Fibonaccizahl.

3.4 Bier in DOS-batch

Als abschließendes Beispiel soll hier noch die Programmierung des Bier Liedes gezeigt werden. Mit Ausnahme einer erweiterten For-Schleife weist dieses Programm keine neuen Element auf. Siehe Listing 4

3.5 Bier in bash

Zum Vergleich wird hier das selbe Programm “99 bottles of beer” als bash script implemetiert. Siehe Listing 5

```
1  @ECHO OFF
2  SET /a number=100
3  FOR /L %%n in (100 -1 1) DO CALL:anzeige %%n
4  GOTO eof
5  :anzeige
6  SET /a x=%1
7  SET /a y=x-1
8  IF %x%==1 GOTO einbier
9  ECHO %x% bottles of beer on the wall, %x% bottles of
       beer.
10 ECHO Take one down and pass it around, %y% bottles of
       beer on the wall.
11 ECHO.
12 GOTO eof
13 :einbier
14 ECHO 1 bottle of beer on the wall, 1 bottle of beer.
15 ECHO Take one down and pass it around, no more bottles
        of beer on the wall.
16 ECHO.
17 ECHO No more bottles of beer on the wall, no more
       bottles of beer.
18 ECHO Go to the store and buy some more, 99 bottles of
       beer on the wall.
```

Listing 4: Bier in DOS-batch

4 Bedeutung und Parallelen zu Linux-bash

Wie bisher zu sehen ist, hat DOS-batch Ähnlichkeiten mit bash in Linux. Beide haben gemeinsame Aufgabengebiete. Auch der starke Zusammenhang mit dem Betriebssystem kann als Gemeinsamkeit gesehen werden. Es muss aber gesagt werden, dass die Mächtigkeit der Linux-bash Sprache eine viel höhere ist, da diese weit tiefer in Hardwarebereiche eingreifen kann. Jedes Batch Programm kann mit wenig Aufwand in ein bash Äquivalent umgeändert werden. Umgekehrt ist dies aus oben erwähnten Gründen nicht der Fall. Auch wenn bis heute eine Eingabeaufforderung und damit DOS-batch in Microsoft Betriebssystemen integriert ist, nimmt die Bedeutung daran ab. Im Gegensatz dazu wäre für einen Linux-Benutzer ein System ohne shell, als das bash oft bezeichnet wird, vollkommend fremd und würde viel von seinem Komfort verlieren. Wenn jemand bash beherrscht, lässt sich sehr viel Arbeit ersparen, bei entsprechnenden Aufgabenstellungen! Die Bedeutung dieser Sprache steigt und nimmt mit zunehmender Verbreitung von Linux als Betriebssystem stetig zu.

```
n=99
bottles=bottles
no
WHILE [ 0 != $[ n ] ]
DO
   ECHO "$n␣$bottles␣of␣beer␣on␣the␣wall,"
   ECHO "$n␣$bottles␣of␣beer,"
   ECHO "take␣one␣down,␣pass␣it␣around,"
   LET n=$n-1
   CASE $n in
   0) no=no
      bottles=${bottles%s}
      ;;
   1) bottles=${bottles%s}
      ;;
   ESAC
   ECHO "${no:-$n}␣$bottles␣of␣beer␣on␣the␣wall."
   ECHO
DONE
```

Listing 5: Bier in bash

5 Zusammenfassung

Die Sprache DOS-batch wurde mit MS-DOS eingeführt und ist auch heute noch immer Bestandteil eines jeden neuen Windowssystems. Ursprünglich diente sie ausschlieslich der Abarbeitung immer wiederkehrender Befehle, wurde aber im Laufe der MS-DOS Entwicklung von Version 1.0 bis 7.22 mit zusätzlichen Funktionen ausgestattet. So wurde beispielsweise eine einfache Arithmetik, Schleifen und Bedingungen eingeführt. Um die Sprache zu verstehen werden anfangs Grundregeln und Grundbefehle sowie Operatoren und Konstrukte näher beleuchtet Abschliesend sollte durch einige Codebeispiele die Bedienung und Programmierung unter DOS-batch erläutert und nähergebracht werden. Zusätzlich sollten kleine Vergleiche mit bash in Linux gezogen werden. Final sollte dem Leser das “Bier-Lied” als Programm aufgezeigt werden. Mithilfe der Dokumentation in dieser Arbeit sollte es möglich sein, diesen Programmcode nachvollziehen zu können.
Die große Verbreitung und Bekanntheit von DOS-Batch ist dem Fakt der festen Zugehörigkeit zu den meist verbreiteten Betriebssystemen Microsoft DOS und Microsoft Windows zuzuschreiben. Auch wenn heute noch einige diese Sprache als absolutes “Pflichtprogramm” für Windows sehen, nimmt der Anteil der Bekanntheit zunehmend ab. DOS-batch kann als eingeschränktes Pendant zu

Shell-Sprachen, wie beispielsweise bash in Linux gesehen werden. Beide sind für ähnliche Aufgaben gedacht, obwohl bash noch weit größere Aufgaben beweltigt, und nicht nur für Dateiorganisation sehr gut geeignet ist.

Literatur

[1] Antoni. Die Geschichte von DOS, 2011. [Online; Stand 16. Februar 2011] http://www.antonis.de/dos/dos-tuts/fischer/5.htm.

[2] Antoni. Großes Tutorial-Paket zur Batch-Programmierung, 2011. [Online; Stand 16. Februar 2011] http://www.antonis.de/dos/batchtut/index.htm.

[3] DosTips. The DOS Batch Guide, 2011. [Online; Stand 16. Februar 2011] http://www.dostips.com/.

[4] Wikipedia. MS-DOS, 2011. [Online; Stand 16. Februar 2011] http://de.wikipedia.org/w/index.php?title=MS-DOS&oldid=85259468.