Interaktive Kommandos unter Windows

Horst Hollatz*, Bianca Truthe†

18. Februar 2003

Zusammenfassung

Die unter Windows existierende Kommandosprache gestattet keine interaktive Kommunikation während der Abarbeitung einer Kommandofolge aus einer `cmd`-Datei. Hier werden 4 Programme vorgestellt, mit deren Hilfe interaktive Kommandos geschrieben werden können, wobei 3 von ihnen hinreichend sind.

Um innerhalb einer Kommandofolge neue Informationen in den Befehlsablauf einfließen zu lassen, muß man Variablen Werte zuweisen, die über die Standardeingabe oder aus einer Datei gelesen werden. Dazu wurde das Programm `READ` geschrieben.
`READ` liest von der Standardeingabe, aus einer Datei oder aus einem String mit dem folgenden Aufruf

```
READ {*var | var [var]} [iz | -sw "s_string"] [-f dateiname]|[-s "string"]
```

`READ` liest zeilenweise vom Eingabemedium. Ist im Aufruf kein Eingabemedium kodiert, wird von der Standardeingabe gelesen. Durch den Wert `iz` bzw. `search_string` wird auf den Anfang der Zeile `iz` bzw. hinter den String `s_string` positioniert, wobei der String zeilenweise gesucht wird (ohne führende Leerzeichen). Beim Lesen aus einem String wird jedes Wort als eine Zeile interpretiert. Beim Lesen von der Standardeingabe darf nicht positioniert werden. Die ab der betreffenden Zeile gelesenen Wörter werden nacheinander den Variablen zugewiesen, wobei bei mehr als einer Variablen der letzten die Restzeile gehört. Falls kein Wort mehr vorhanden ist, erhält die Variable das Leerzeichen zugewiesen. Falls der 1. Variablen ein `*` vorangestellt ist, erhält sie alle Wörter. Ein Aufruf mit dem alleinigen Parameter `!` wirkt als Leeranweisung. Einen solchen Aufruf benötigt man z. B., um den Wert einer Variable zu reproduzieren; er ist gegebenenfalls auch nötig, um Variablenwerte in die laufende Shell zu holen, z. B. nach Abschluß einer Pipe. Es wird keine vollständige Syntaxprüfung des Aufrufs vorgenommen; insbesondere müssen die verwendeten Variablennamen gültig sein.
Wichtige Bemerkungen:

1. Die Umgebungsvariable `tmp` muß auf einen (temporären) Ordner zeigen, andernfalls wird der aktuelle Ordner benutzt.

*http://horst.hollatz.de
†http://panda.math.uni-magdeburg.de

2. Vor der ersten Verwendung muß `read.exe` einmal parameterlos aufgerufen werden. Dies kann z. B. am Beginn der Kommandofolge geschehen. Dabei wird im temporären bzw. aktuellen Ordner eine `cmd`-Datei angelegt, die zunächst nur eine Zeile enthält:

   ```
   @echo off & call read.exe &1 &2 &3 &4 &5 &6 &7 &8 &9
   ```

3. Das Programm `read` darf danach nur implizit durch

   ```
   call %tmp%\read.cmd parametersatz
   ```

   bzw.

   ```
   call read.cmd parametersatz
   ```

   aufgerufen werden. Es dürfen höchstens 5 Variable im Aufruf auftreten.

4. Für jeden im Aufruf stehenden Variablennamen `va` erzeugt das Programm `read.exe` in der Datei `read.cmd` eine Zeile `set va=` mit den entsprechenden Werten. Diese Zeilen werden nach Verlassen der `exe`-Datei abgearbeitet. Da auch die `cmd`-Datei mit `call` aufgerufen wurde, kann man nach Rückkehr aus der aufgerufenden `cmd`-Datei über die Werte der Variablen verfügen.

Beispiel:

```
@echo off & read.exe
call %tmp%\read.cmd a
```

Durch den Aufruf von `read.cmd` wird eine Eingabe verlangt, um diese der Variablen `a` zuzuweisen; in folgenden Befehlen kann dieser Wert benutzt werden.
Beispiel:

```
@echo off & read.exe
call %tmp%\read.cmd a b 7 -f xx.txt
call %tmp%\read.cmd c d e 2 -f xx.txt
```

Die Variable `a` erhält aus der 7. Zeile von `xx.txt` das 1. Wort zugewiesen; `b` erhält den Rest der Zeile; `c` erhält das erste Wort der 2. Zeile , `d` das 2. Wort und `e` den Rest der Zeile.

Häufig benötigt man in einer Befehlsfolge Teilwörter von vorhandenen, wie z. B. den Ordner, in dem sich eine gewisse Datei befindet oder den Dateityp. Man muß dazu Teilwörter aus anderen herausschneiden und das Ergebnis einer Variablen zuweisen. Dazu dient das Programm `CUT`:

```
CUT var op string
```

dabei ist `var` der Name jener Variablen, die das Resultat der Beschneidung aufnimmt.
Hat `op` die Form `*w`, so wird von vorn der kürzeste, mit `w` endende Teilstring abgeschnitten; im Falle `**w` ist es der längste Teilstring von vorn. Entsprechendes bedeutet `w*`, daß

der kürzeste, mit `w` beginnende Teilstring von hinten abgeschnitten wird und bei `w**` ist es der längste. Falls die Teilstring-Suche nicht erfolgreich ist, wird der String unverändert übergeben. Durch Beispiele wird alles klar.
Beispiel:

```
CUT a **. C:\tmp\hi.a\w.txt
```

Hier wird von vorn der längste Teilstring abgeschnitten, der mit einem Punkt endet; also enthält die Variable `a` nach dem Aufruf den Dateityp.
Beispiel:

```
CUT a *\ C:\tmp\hi.a\w.txt
```

Es wird von vorn alles bis zum ersten Backslash abgeschnitten; die Variable hat danach den Wert `tmp\hi.a\w.txt`.
Beispiel:

```
CUT a :** C:\tmp\hi.a\w.txt
```

Es wird von hinten der längste Teilstring abgeschnitten, der mit : beginnt. Um den Aufruf für Standardfälle lesbarer zu machen, gibt es für `op` die zusätzlichen Wörter `-drive, -dirname, -filename, -basename, -type`, die sich selbst erklären dürften.

Innerhalb eines interaktiven Kommandos benötigt man oft auch eine ganzzahlige Arithmetik mit Variablen, deren Werte ganze Zahlen sind. Das Programm `EXPR` berechnet arithmetische Ausdrücke, um sie einer Variablen in einer `cmd`-Datei zuzuweisen. Es hat den folgenden Aufruf:

```
EXPR var val1 op val2,
```

wobei `var` der Variablenname, `val1` der 1. Operand, `val2` der 2. Operand und `op` eine der Operationen aus {`+, -, *, /`} bedeuten. Die äußeren Bedingungen sind analog zu oben.
Beispiel:

```
@echo off & expr.exe
set a=2
call %tmp%\expr.cmd a %a% + 5
call %tmp%\expr.cmd b 17 - %a%
```

Nach dem Aufruf haben `a` den Wert 7 und `b` den Wert 10.
Ein einfaches, zusammenhängendes Beispiel soll die Funktionsweise illustrieren. Dazu sei die Aufgabe gestellt, interaktiv die Dateien eines Ordners zu löschen. Das folgende Kommando leistet das Verlangte.

```
@echo off & read.exe & cut.exe & expr.exe
echo Loeschen von Dateien in einem Ordner
echo Bitte einen Ordnernamen mit Pfad eingeben:
call %tmp%\read.cmd d
if %d%==. goto anf
call %tmp%\cut.cmd lw -drive %d%
```

```
   if not %lw%==%d% %lw%:
   if not exist %d% goto ende
   cd %d%
:anf
   dir /a-d /-w /b /l | %tmp%\read.cmd *a
   call %tmp%\read.cmd !
   set i=1
:anfang
   call %tmp%\read.cmd b %i% -s "%a%"
   if "%b%"==" " goto ende
   echo %b% loeschen?(j/n,anderes=quit)
   call %tmp%\read.cmd c
   if %c%==n goto weiter
   if not %c%==j goto ende
   del %b%
:weiter
   call %tmp%\expr.cmd i %i% + 1
   goto anfang
:ende
   pause
```

In diesem Kommando werden zunächst die Namen aller Dateien aus dem angegebenen Ordner ermittelt und auf der Variablen `a` abgelegt, um sodann nacheinander jeden Dateinamen anzuzeigen und zu fragen, ob die betreffende Datei zu löschen ist. Die Antwort `j` führt zum Löschen der betreffenden Datei. Bei Eingabe eines von `j` oder `n` verschiedenen Wortes endet das Kommando sofort.

Wendet man dieses Kommando auf einen Ordner mit vielen Dateien an, wünscht man sich sicher ein Programm, das die Auswahl der zu löschenden Dateien beschleunigt. Dazu gibt es das Programm `SEL`, mit dem man Wörter aus einer Liste auswählen kann. Der Aufruf lautet:

```
SEL {*var | var} "w1 ... wn"
```

Nach dem Aufruf erscheinen die Auswahlwörter `w1` bis `wn` aufsteigend numeriert auf dem Bildschirm mit der Aufforderung, eine Auswahl zu treffen. Es dürfen Zahlen in der Form `i i-k` mit Wiederholung eingegeben werden. Die entsprechenden Wörter werden der Variablen `var` zugewiesen, falls `*var` im Aufruf codiert wurde; andernfalls erhält `var` nur das erste ausgewählte Wort. Im Falle der Eingabe von `<` hat `var` das Leerzeichen als Wert. Bei langen und mehr als 8 Auswahlwörtern wird versucht, die Auswahlwörter für die Anzeige zu kürzen, indem sie bei einem Backslash gebrochen werden.
Mit diesem Programm erhält das obige Beispiel die folgende Form.

```
   @echo off & read.exe & sel.exe & cut.exe
   echo Loeschen von Dateien in einem Ordner
   echo Bitte einen Ordnernamen mit Pfad eingeben:
   call %tmp%\read.cmd d
   if %d%==. goto anf
   call %tmp%\cut.cmd lw -drive %d%
   if not %lw%==%d% %lw%:
   if not exist %d% goto ende
   cd %d%
:anf
   dir /a-d /-w /b /l | %tmp%\read.cmd *a
   call %tmp%\read.cmd !
```

```
   call %tmp%\sel.cmd *a "%a%"
   if "%a%"==" " goto ende
   echo Sollen die folgenden Dateien wirklich geloescht werden?(j/n)
   echo %a%
   call %tmp%\read.cmd b
   if not %b%==j goto ende
   del %a%
:ende
   pause
```

Die Programme sind mit `Visual C++` übersetzt worden.
Kostenfreie Bezugsquelle.