

HOWTO zur Shell-Kommando-Kombination

(C) 2006 T.Birnthaler/H.Gottschalk <howtos(at)ostc.de>

OSTC GmbH, <http://www.ostc.de>

\$Id: shell-command-combination-HOWTO.txt,v 1.5 2008-09-16 09:45:05 tsbirn Exp \$

Dieses Dokument beschreibt die verschiedenen Verfahren zur Kombination von Shell-Kommandos.

## Inhaltsverzeichnis

- 1) Kommandos HINTEREINANDER ausführen
- 2) Hintergrund-Kommandos
- 3) Pipe
- 4) Kommando-Substitution
- 5) UND-Verknüpfung
- 6) ODER-Verknüpfung
- 7) Subshell
- 8) Gruppierung

Die Shell erlaubt nur EIN Kommando pro Zeile, jedes Kommando muss durch einen Zeilenvorschub abgeschlossen werden (oder durch einen ";"). Allerdings gibt es eine Vielzahl von Kommando-Kombinationen mit Hilfe bestimmter Sonderzeichen, durch die auch zwei (oder mehr) Kommandos auf einer Zeile erlaubt ist:

### 1) Kommandos HINTEREINANDER ausführen

Erst CMD1, nach dessen Abschluß CMD2, die Kommandos sind NICHT miteinander verknüpft:

```
CMD1          oder          CMD1; CMD2
CMD2
```

### 2) Hintergrund-Kommandos

Kommandos GLEICHZEITIG (parallel) im Hintergrund ausführen, die Kommandos sind NICHT miteinander verknüpft:

```
CMD1 &          oder          CMD1 & CMD2 &
CMD2 &
```

### 3) Pipe

Kommandos GLEICHZEITIG (parallel) starten und die Standard-Ausgabe von Kommando CMD1 an die Standard-Eingabe von Kommando CMD2 übergeben. Die beiden Prozesse synchronisieren sich über den von der Pipe "|" bereit gestellten Speicher (etwa 2-16 KByte), indem Kommando CMD1 nur dorthin schreibt, wenn CMD2 Daten empfangen kann und CMD2 nur daraus liest, wenn CMD1 Daten dorthin geschrieben hat:

```
CMD1 | CMD2          auch          CMD1 | CMD2 | ... | CMD_N
```

### 4) Kommando-Substitution

Zuerst das Kommando CMD2 ausführen und sein Ergebnis die Kommandozeile von CMD1 einfügen und anschließend Kommando CMD1 aufrufen:

```
CMD1 `CMD2`          # Alte Form
CMD1 $(CMD2)          # Moderne Form (nur bash und ksh)
```

### 5) UND-Verknüpfung

Nur dann Kommando CMD2 ausführen, wenn Kommando CMD1 erfolgreich ablief (d.h. einen Exit-Status von "0" ergab).

```
CMD1 && CMD2          # Bsp: [ -e FILE ] && rm FILE
```

### 6) ODER-Verknüpfung

Nur dann Kommando CMD2 ausführen, wenn Kommando CMD1 NICHT erfolgreich ablief (d.h. einen Exit-Status ungleich "0" ergab):

```
CMD1 || CMD2          # Bsp: [ -n "$VAR" ] || VAR=Default
```

### 7) Subshell

Die Kommandos in einer gemeinsamen SUBSHELL hintereinander starten, die Ausgaben können gemeinsam umgelenkt werden:

```
( CMD1; CMD2 )          oder          ( CMD1; CMD2 ) > out 2> err
```

## 8) Gruppierung

-----

Die Kommandos in der aktuellen Shell hintereinander starten und die Ausgaben der Kommandos zusammenfassen (können gemeinsam umgelenkt werden):

```
{ CMD1; CMD2; }      # Strichpunkt nach dem letztem Kommando nötig!  
                    # Leerzeichen nach "{" und vor "}" notwendig!
```