(C) 2024 OSTC GmbH (http://www.ostc.de)

HOWTO zur Shell-Kommando-Kombination

(C) 2006-2017 T.Birnthaler/H.Gottschalk <howtos(at)ostc.de>
OSTC Open Source Training and Consulting GmbH
http://www.ostc.de

$Id: shell-command-combination-HOWTO.txt,v 1.16 2019/11/26 19:37:07 tsbirn Exp $

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

INHALTSVERZEICHNIS

0) ÃM-^\bersicht
1) Kommandos nacheinander ausfÃ¼hren
2) Hintergrund-Kommandos
3) Pipe
4) Kommando-Substitution
5) UND-VerknÃ¼pfung
6) ODER-VerknÃ¼pfung
7) Subshell
8) Gruppierung/Codeblock
9) TemporÃ¤re Datei

Shells (sh, bash, ksh, zsh, csh, tcsh, ...) erlauben nur EIN Kommando pro Zeile, jedes Kommando muss durch einen Zeilenumbruch abgeschlossen werden (oder durch einen ";"). Allerdings gibt es eine Vielzahl von Kommando-Kombinationen mit Hilfe bestimmter Sonderzeichen, durch die auch zwei (oder mehr) Kommandos in einer Zeile erlaubt sind.

0) ÃM-^\bersicht

Die Kommando-Kombinationen lassen sich folgendermaÃM-^_en charakterisieren:

- Kommandos: Laufen parallel oder sequentiell ab
- Kommandos: Laufen abhÃ¤ngig oder unabhÃ¤ngig voneinander ab
- Kommandos: Gemeinsam umlenkbar/in Hintergrund stellbar oder nicht
- Subshell: Zur AusfÃ¼hrung zusÃ¤tzlich gestartet oder nicht
- Exit-Status: Abfragbar oder nicht
- Shell-Variablen: Gemeinsam oder nicht (nur Shell-Kommandos)
- Umgebungs-Variablen (Environment-Var.): Gemeinsam oder nicht (Environment) (an Kindprozess immer vererbt, nicht zurÃ¼ckgegeben)

Kombination	Parallel	AbhÃ¤ngig	Gem.	Subsh	Exit	ShVar	EnvVar
NEWLINE ;	NEIN	NEIN	NEIN	NEIN	JA	JA	JA
&	JA	NEIN	NEIN	JA	NEIN	NEIN	NEIN
|	JA	JA	NEIN	JA	JA	NEIN	NEIN
`...` $(...)	NEIN	JA	NEIN	JA	JA	NEIN	NEIN
&&	NEIN	JA	NEIN	NEIN	JA	JA	JA
||	NEIN	JA	NEIN	NEIN	JA	JA	JA
(...)	NEIN	NEIN	JA	JA	JA	NEIN	NEIN
{...}	NEIN	NEIN	JA	NEIN	JA	JA	JA
<(...)	NEIN	NEIN	--	JA	NEIN	NEIN	NEIN

HINWEIS: Nach Pipe-Symbol "|", logischer VerknÃ¼pfung "&&" und "||" sowie nach "{" darf ein Zeilenumbruch folgen. Ebenso darf innerhalb der Hochkommas "..." und '...' sowie `...` und der Klammern "(...)" und "{...}" beliebig Zeilenumbruch verwendet werden.

1) Kommandos nacheinander ausfÃ¼hren

Erst CMD1 ausfÃ¼hren und nach dessen AbschluÃM-^_ CMD2 ausfÃ¼hren, die Kommandos sind NICHT miteinander verknÃ¼pft:

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

Direkt nach jedem Kommando kann sein Exit-Status abgefragt werden:

```
CMD1          oder        CMD1; echo "$?"
echo "$?"
```

Da die Abfrage ein neues Kommando darstellt, ist der Exit-Status anschlieÃM-^_end

```
Ã¼berschrieben. Soll er mehrfach verwendet werden, ist er zwischenzuspeichern:

  ES="$?"


2) Hintergrund-Kommandos
------------------------
Kommandos GLEICHZEITIG (parallel) im Hintergrund ausfÃ¼hren, die Kommandos sind
NICHT miteinander verknÃ¼pft:

  CMD1 &        oder        CMD1 & CMD2 &
  CMD2 &

Der Exit-Status eines Hintergrund-Kommandos kann NICHT abgefragt werden. Die
Shell liefert immer Exit-Status 0 (OK) zurÃ¼ck, falls das Kommando gestartet
werden konnte. Falls das Kommando nicht gestartet werden konnten, liefert sie
Fehlerstatus 126 oder 127 zurÃ¼ck.


3) Pipe
-------
Kommandos GLEICHZEITIG (parallel) starten und die Standard-Ausgabe von Kommando
CMD1 an die Standard-Eingabe von Kommando CMD2 Ã¼bergeben. Die beiden Prozesse
synchronisieren sich Ã¼ber den von der Pipe "|" bereit gestellten Puffer im
Speicher (etwa 2-16 KByte), indem Kommando CMD1 nur dorthin schreibt, wenn
Platz im Puffer vorhanden ist und CMD2 nur daraus liest, wenn Daten im Puffer
vorhanden sind:

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

Der Exit-Status der Pipeline ist der Exit-Status des LETZTEN Kommandos. Falls
ein Kommando mit Fehler abbricht, werden alle Kommandos der Pipeline beendet.


4) Kommando-Substitution
------------------------
Zuerst das Kommando CMD2 ausfÃ¼hren, seine Ausgabe auf der Standard-Ausgabe
in die Kommandozeile von CMD1 einfÃ¼gen und dann Kommando CMD1 aufrufen:

  CMD1  `CMD2`    # Alte Form    (Bourne-Shell sh)
  CMD1 $(CMD2)    # Moderne Form (nur bash und ksh)

Nur der Exit-Status von CMD1 kann abgefragt werden. Um den Exit-Status von
CMD2 und CMD1 abzufragen, folgende Konstruktion benutzen:

  RESULT=$(CMD2)    # Ausgabe von CMD2 abfangen
  echo "$?"         # Exit-Status von CMD2
  CMD1 "$RESULT"    # Ausgabe von CMD2 an CMD1 Ã¼bergeben
  echo "$?"         # Exit-Status von CMD1


5) UND-VerknÃ¼pfung
------------------
Nur dann Kommando CMD2 ausfÃ¼hren, wenn Kommando CMD1 erfolgreich ablief
(d.h. einen Exit-Status von "0" = Ok ergab):

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

Exit-Status ist der Exit-Status des LETZTEN von links nach rechts ausgefÃ¼hrten
Kommandos (ein Exit-Status ungleich 0 bricht die Verarbeitung ab).


6) ODER-VerknÃ¼pfung
-------------------
Nur dann Kommando CMD2 ausfÃ¼hren, wenn Kommando CMD1 NICHT erfolgreich ablief
(d.h. einen Exit-Status ungleich "0" = Fehler ergab):

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

HINWEIS: && und || NICHT mischen, da die Bedeutung derartiger VerknÃ¼pfungen
sehr schwer verstÃ¤ndlich ist (obwohl sie klar festgelegt ist):

  CMD1 && CMD2 || CMD3   # CMD1 Fehler? -> CMD2+CMD3 nicht ausfÃ¼hren
                         # CMD1 Ok?     -> CMD2 Ok?     -> CMD3 nicht ausfÃ¼hren!
                         # CMD1 Ok?     -> CMD2 Fehler? -> CMD3 ausfÃ¼hren!
  CMD1 || CMD2 && CMD3   # CMD1 Ok?     -> CMD2+CMD3 nicht ausfÃ¼hren
                         # CMD1 Fehler? -> CMD2 Fehler? -> CMD3 nicht ausfÃ¼hren!
                         # CMD1 Fehler? -> CMD2 Ok?     -> CMD3 ausfÃ¼hren!

Exit-Status ist der Exit-Status des LETZTEN von links nach rechts ausgefÃ¼hrten
Kommandos (ein Exit-Status gleich 0 bricht die Verarbeitung ab).
```

(C) 2024 OSTC GmbH (http://www.ostc.de)


```
7) Subshell
-----------
Kommandos CMD1 und CMD2 in einer gemeinsamen SUBSHELL hintereinander starten,
die Ausgaben kÃ¶nnen gemeinsam umgelenkt werden:

  ( CMD1; CMD2 )                # Gemeinsam in Subshell starten
  ( CMD1; CMD2 ) > out 2> err   # Gemeinsam Ausgabe umlenken
  ( CMD1; CMD2 ) &              # Gemeinsam in Hintergrund schicken

Exit-Status ist der Exit-Status des letzten in der Klammer ausgefÃ¼hrten
Kommandos.


8) Gruppierung/Codeblock
------------------------
Kommandos CMD1 und CMD2 in der aktuellen Shell hintereinander starten und die
Ausgaben der Kommandos zusammenfassen (kÃ¶nnen gemeinsam umgelenkt werden) oder
sie gemeinsam im Hintergrund ausfÃ¼hren.

  { CMD1; CMD2; }                # Analog CMD1; CMD2; ohne {...}
  { CMD1; CMD2; } > out 2> err   # Gemeinsam Ausgabe umlenken (in akt. Shell)
  { CMD1; CMD2; } &              # Im Hintergrund: Erzeugt trotzdem eine Subshell

Exit-Status ist der Exit-Status des letzten in der Klammer ausgefÃ¼hrten
Kommandos.

ACHTUNG: Ein Strichpunkt ";" ist auch nach dem letzten Kommando notwendig und die
Leerzeichen nach "{" und vor "}" sind ebenfalls notwendig!

HINWEIS: Geschweifte Klammern "{" und "}" sind aus historischen GrÃ¼nden
Shell-SchlÃ¼sselworte --> Werden nur als 1. Symbol einer Zeile oder nach ";"
erkannt und Leerzeichen davor/dahinter sind notwendig!


9) TemporÃ¤re Datei
------------------
Die Ergebnisausgabe eines Kommandos als temporÃ¤re Datei ablegen und einem
anderen Kommando in Form eines Dateinamens Ã¼bergeben:

  CMD1 <(CMD2)

Nur der Exit-Status von CMD1 kann abgefragt werden. Um den Exit-Status von
CMD2 und CMD1 abzufragen, folgende Konstruktion benutzen:

  RESULT=$(CMD2)          # Ausgabe von CMD2 abfangen
  echo "$?"               # Exit-Status von CMD2
  echo "$RESULT" | CMD1   # Ausgabe von CMD2 an CMD1 Ã¼bergeben
  echo "$?"               # Exit-Status von CMD1
```