

HOWTO zu Shell- und Umgebungs/Environment-Variablen

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

\$Id: shell-variable-HOWTO.txt,v 1.17 2017/09/11 22:59:53 tsbirn Exp \$

Dieses Dokument beschreibt die Syntax und die Eigenschaften von Shell- und Umgebungs-Variablen der Shell-Familien "sh" ("bash", "ksh", "zsh") und "csh" ("tcsh").

INHALTSVERZEICHNIS

- 1) Shell-Variablen in der "sh"-Familie
- 2) Umgebungs/Environment-Variablen in der "sh"-Familie
- 3) Bedingte Bewertung von Shell-Variablen
- 4) Beispiele
 - 4.1) Variable PATH erweitern
 - 4.2) Variable PATH verkürzen
 - 4.3) Beispiel-Skript var.sh (mit Zeilennummern)
- 5) Variablen in der "csh"-Familie
- 6) Syntax/Eigenschaften-Vergleich der Variablen in "sh" und "csh"

1) Shell-Variablen der "sh"-Familie

Shell-Variablen sind Paare der Form (NAME=WERT), jede Shell verwaltet in ihrem Datenspeicher eine (beliebig lange) Liste davon. Diese Variablen steuern das Verhalten der Shell oder aus der Shell heraus aufgerufener Programme. Neben einer Reihe von vordefinierten Standardvariablen kann der Benutzer beliebige weitere Shell-Variablen anlegen (und auch wieder löschen). Die wichtigsten Shell-Variablen sind:

Name	Beschreibung
CDPATH	Suchpfad für Kommando "cd"
EDITOR	Definiert den Standard-Editor (vi/emacs)
HOME	Standardverzeichnis für "cd" (Home-Verzeichnis)
IFS	Whitespace-Zeichen für read-Befehl ("internal field separator")
LOGNAME	Aktueller Loginname (auf manchen UNIX-Systemen nicht vorhanden)
MANPATH	Suchpfad für man-Kommando
PATH	Suchpfad für Kommandoaufruf
PS1	Shell-Prompt ("\$ ")
PS2	Fortsetzungs-Prompt ("> ")
PWD	Aktuelles Verzeichnis ("print working directory")
RANDOM	Zufallszahl (0..32767)
SHELL	Name der Login-Shell
TERM	Terminaltyp (für Editoren, more/less, curses-Bibliothek)
TZ	Zeitzone ("time zone")
USER	Aktueller Loginname (auf manchen UNIX-Systemen nicht vorhanden)
VISUAL	Definiert den Standard-Editor (vi/emacs)

Die Kommandos zum Setzen, Anzeigen, Verwenden und Löschen von Shell-Variablen lauten:

Befehl	Beschreibung
VAR=TEXT	Erzeugt eine Shell-Variable (keine Leerzeichen um "="!)
\$VAR	Zugriff auf den Wert (Inhalt) einer Shell-Variablen
\${VAR}xxx	Sichere Verwendung, falls Text "xxx" direkt dahinter steht
VAR=	Löschen einer Shell-Variablen (anschließend leer)
unset VAR	Löschen einer Shell-Variablen (anschließend undefiniert!=leer)
set	Alle Shell-Variablen auflisten

2) Umgebungs/Environment-Variablen der "sh"-Familie

Ein spezieller Typ von Shell-Variablen sind die sogenannten "Umgebungs/Environment-Variablen", sie sind eine TEILMENGE der Shell-Variablen. JEDER Prozess besitzt eine Liste von Umgebungsvariablen (nicht nur die Shell). Sie werden beim START jedes Kindprozessen vom Elternprozess an diesen "vererbt" (eine Shell macht dies ebenfalls, vererbt allerdings nicht ihre Shell-Variablen). Die Kindprozesse können die vererbte Variablen-Liste anschließend beliebig verändern und erweitern und diese geänderte Liste

ihrerseits an eigene Kindprozesse weitervererben. In den Elternprozess "zurückschreiben" können Kindprozesse ihren Umgebungsbereich hingegen nicht.

Befehl	Beschreibung
export VAR	Shell-Variablen in Umgebungs-Variable umwandeln
export VAR=TEXT	Analog + gleichzeitige Wertzuweisung (nur "bash")
\$VAR	Zugriff auf den Wert (Inhalt) einer Shell-Variablen
\${VAR}	Sichere Verwendung, falls Text direkt dahinter steht
VAR=	Löschen einer Umgeb.-Variablen (anschließend leer)
unset VAR	Löschen einer Umgeb.-Variablen (anschließend undefiniert)
env	Alle Umgebungs-Variablen auflisten
printenv	Alle Umgebungs-Variablen auflisten

HINWEIS:

- * Bei der Zuweisung sind KEINE Leerzeichen um das "=" erlaubt.
- * Konvention: In der "sh" werden alle Variablen GROSS geschrieben (wahrscheinlich um sie besser erkennen zu können, da alles andere klein geschrieben wird).
- * Shell-Variablen sind nur für die Shell selbst relevant, externe Kommandos sehen sie NICHT.
- * Umgebungs/Environment-Variablen sind für EXTERNE Kommandos relevant, sie werden automatisch an von der Shell aufgerufene externe Kommandos weitergegeben.
- * Eine Shell-Variable kann mit "export" in eine Umgebungs-Variable umgewandelt werden (die umgekehrte Richtung ist nur möglich durch löschen mit "unset" und Neudefinition der Variablen).
- * Die Reihenfolge von Zuweisung eines Werts zu einer Variablen und exportieren der Variablen ist egal. In der "bash" können diese beiden Anweisungen sogar in einer Anweisung kombiniert werden.
- * Shell-Variablen werden auch als "lokale" Variablen bezeichnet (lokal zur Shell).
- * Umgebungs-Variablen werden auch als "globale" Variablen bezeichnet, eine bessere Bezeichnung wäre "vererbte" Variablen (an alle Subshells einer Shell).
- * Für die Ausführung von Shell-Skripten (Kommando/Batch-Prozeduren) wird immer eine "Sub-Shell" (d.h. ein Kindprozess) gestartet.
- * Aliase und Funktionen werden grundsätzlich nicht an Sub-Shells "vererbt".

3) Bedingte Bewertung von Shell-Variablen

Um auf den Wert einer Shell-Variablen zuzugreifen, sind neben den Standardformen "\$VAR" bzw. "\${VAR}" noch folgende Varianten möglich:

Zugriff	Bedeutung
\${VAR-TEXT}	VAR zurück, falls VAR definiert; sonst TEXT zurück
\${VAR=TEXT}	Analog + Zuweisung von TEXT an VAR
\${VAR+TEXT}	TEXT zurück, falls VAR definiert; sonst nichts zurück
\${VAR?}	Ausgabe "VAR: parameter null or not set"
	+ Abbruch falls VAR undefiniert; sonst VAR zurück
\${VAR?TEXT}	Analog Ausgabe von "VAR: TEXT"; sonst VAR zurück
`\${VAR:-TEXT}	VAR zurück, falls VAR leer; sonst TEXT zurück
`\${VAR=TEXT}	Analog + Zuweisung von TEXT an VAR
`\${VAR+TEXT}	TEXT zurück, falls VAR leer; sonst nichts zurück
`\${VAR:?}	Ausgabe "VAR: parameter null or not set"
	+ Abbruch falls VAR leer; sonst VAR zurück
`\${VAR:?TEXT}	Analog Ausgabe von "VAR: TEXT"; sonst VAR zurück

HINWEIS:

- * Ein Doppelpunkt nach VAR verlangt, dass die Variable VAR nicht LEER sein darf ("not null"). Ohne Doppelpunkt muß sie DEFINIERT sein (darf aber leer sein).

4) Beispiele

4.1) Variable PATH erweitern

```

echo $PATH           Inhalt von Variable PATH anzeigen
PATH=               Variable PATH löschen
PATH="/bin:/usr/bin" Variable PATH setzen ("..." auch weglassbar)
PATH="$PATH:."      Variable PATH hinten um "." erweitern
echo $PATH          => /bin:/usr/bin:.
PATH="$HOME:$PATH" Variable PATH vorn um "$HOME" erweitern
echo $PATH          => /home/user33:/bin:/usr/bin:.

```

4.2) Variable PATH verkürzen

```

echo $PATH          => /home/user33:/bin:/usr/bin:.
PATH=`echo $PATH | Verzeichnis "/usr/bin" aus PATH entfernen
  sed "s#^/usr/bin:##" | ...kann am Anfang
  sed "s#:/usr/bin:##" | ...in der Mitte
  sed "s#:/usr/bin$##" ` ...oder am Ende vorkommen
echo $PATH          => /home/user33:/bin:.
PATH=$(echo $PATH | Analog mit Bash-Syntax $(...) statt `...`
  sed "s#^/usr/bin:##" |
  sed "s#:/usr/bin:##" |
  sed "s#:/usr/bin$##")

```

4.3) Beispiel-Skript var.sh (mit Zeilennummern)

```

[ 1] # Übergebene Werte ausgeben
[ 2] echo "SHVAR=$SHVAR"
[ 3] echo "ENVAR=$ENVAR"
[ 4]
[ 5] # Werte neu belegen
[ 6] SHVAR="wert1"
[ 7] ENVAR="wert2"
[ 8]
[ 9] # Neu belegte Werte ausgeben
[10] echo "SHVAR=$SHVAR"
[11] echo "ENVAR=$ENVAR"

```

Ausführung und Ausgabe des Skriptes var.sh:

```

$ SHVAR=sss          # Variable SHVAR in Login-Shell belegen
$ ENVAR=eee          # Variable ENVAR in Login-Shell belegen
$ export ENVAR       # ENVAR ist Umgebungs-Variable (wird "vererbt")
$ echo $SHVAR $ENVAR # Variableninhalt in Login-Shell ausgeben
sss eee              # => Ergebnis
$ sh var.sh          # Skript "var.sh" aufrufen (--> Sub-Shell)
# Ausgabe:
SHVAR=               # --> Shell-Variable ist leer, da nicht "vererbt"
ENVAR=eee            # --> Umgebungs-Variable ist "vererbt" worden
SHVAR=wert1          # --> Variableninhalt in Sub-Shell
ENVAR=wert2          # --> Variableninhalt in Sub-Shell
# Skriptende
$ echo $SHVAR $ENVAR # Variableninhalt in Login-Shell ausgeben
sss eee              # --> Ergebnis

```

5) Variablen in der "csh"-Familie

Die Kommandos zum Setzen, Anzeigen, Verwenden und Löschen von Shell- und Umgebungs-Variablen in der "csh"-Familie lauten:

Befehl	Beschreibung
set VAR	Erzeugt eine leere Shell-Variable
@ VAR	Analog
set VAR = TEXT	Analog + gleichzeitige Wertzuweisung
@ VAR = TEXT	Analog
\$VAR	Zugriff auf den Wert (Inhalt) einer Shell-Variablen
\${VAR}	Sichere Verwendung, falls Text direkt dahinter steht
set VAR =	Löschen einer Shell-Variablen (anschließend leer)
unset VAR	Löschen einer Shell-Variablen (anschließend undefiniert)
set	Alle Shell-Variablen auflisten
@	Analog
setenv VAR	Erzeugt eine Umgebungs-Variable
setenv VAR TEXT	Analog + gleichzeitige Wertzuweisung
\$VAR	Zugriff auf den Wert (Inhalt) einer Umgebungs-Variablen
\${VAR}	Sichere Verwendung, falls Text direkt dahinter steht

setenv VAR	Löschen einer Umgebungs-Variablen (anschließend leer)	
unsetenv VAR	Löschen einer Umgebungs-Variablen (anschließend undef.)	
env	Alle Umgebungs-Variablen auflisten	
printenv	Alle Umgebungs-Variablen auflisten	
+-----+		

- * In der "csh" sind die Umgebungs-Variablen und die Shell-Variablen GETRENNTE BEREICHE, eine Umwandlung der einen Sorte in die andere ist nicht möglich.
- * In der "csh" gibt es einige Variablen, die bis auf die Groß/Kleinschreibung gleich heißen und auch den gleichen Wert enthalten: HOME/home, PATH/path, TERM/term, PWD/cwd, SHELL/shell, USER/user. Bei Änderung der Shell-Variablen ändert sich (meist) auch die Umgebungs-Variable, bei Änderung der Umgebungs-Variablen bleibt die Shell-Variable unverändert:

```

set path = xxx # --> PATH = xxx
set home = xxx # --> HOME = xxx
set user = xxx # --> USER = xxx
set term = xxx # --> TERM = xxx
set cwd = xxx # --> PWD bleibt
set shell = xxx # --> SHELL bleibt

```

6) Syntax/Eigenschaften-Vergleich der Variablen in "sh" und "csh"

Folgende Tabelle vergleicht die Syntax und die Eigenschaften von Shell- und Umgebungs-Variablen Shell-Familien "sh" und "csh":

Typ	Shell-Familie	
	sh (bash, ksh)	csh (tcsh)
Shell-Variable	VAR= VAR=WERT set	set var = @ var = set var = WERT @ var = WERT set @
	echo \$VAR unset VAR	echo \$var unset var
Shell-Array	VAR=(WERT1 WERT2...) (bash!) VAR=([0]=W1 [2]=2...) (bash!) VAR[N]=WERTn (nur bash!) echo \${VAR[N]} (nur bash!) echo \${VAR[*]} (nur bash!) echo \${VAR[@]} (nur bash!) echo \$#VAR[*] (nur bash!)	set var = (WERT1 WERT2 ...) echo \$var[N]
Umgebungs-Variable / Environ-ment-Var.	VAR= VAR=WERT export VAR export VAR=WERT (nur bash!)	setenv VAR setenv VAR WERT setenv VAR \${WERT1}:\${WERT2}:...
	env echo \$VAR unset VAR	(print)env echo \$VAR unsetenv VAR

HINWEIS:

- * In der "sh" sind bei der Zuweisung KEINE Leerzeichen um das "=" erlaubt, In der "csh" sind Leerzeichen erlaubt, dürfen aber auch fehlen.
- * In der "csh" gelten folgende Konventionen:
 - + Shell-Variable: klein geschrieben
 - + Umgebungs-Variable: GROSS geschrieben.
- * In der "sh" sind die Umgebungs-Variablen eine UNTERMENGE der Shell-Variablen, eine Shell-Variable kann mit "export" in eine Umgebungs-Variable umgewandelt werden (umgekehrte Richtung nur durch Löschen mit "unset" + Neudefinition).
- * In der "csh" sind die Umgebungs-Variablen und die Shell-Variablen GETRENNTE BEREICHE, eine Umwandlung der einen Sorte in die andere ist nicht möglich.
- * Array-Variablen sind in der "bash" und in der "csh" möglich.
 - + "bash": Indizierung mit numerischem Index ab 0 sowie mit Texten (assoziativ/Hash)

+ "csh": Indizierung mit numerischem Index ab 1.