HOWTO zu Shell- und Umgebungs/Environment-Variablen

(C) 2006-2024 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 1.19 2025/02/18 10:08:31 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 in 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 Standard-Variablen kann der Benutzer beliebige weitere Shell-Variablen anlegen (und auch wieder löschen). Die wichtigsten Shell-Variablen sind:

Name	Beschreibung
HOME	Standard-Verzeichnis für "cd" (Home-Verzeichnis)
PWD	Aktuelles Verzeichnis [print working directory]
PATH	Suchpfad für Kommandoaufruf (durch ":" getrennte Verz.liste)
CDPATH	Suchpfad für cd-Kommando (durch ":" getrennte Verz.liste)
MANPATH	Suchpfad für man-Kommando (durch ":" getrennte Verz.liste)
USER	Aktueller Loginname (auf manchen UNIX-Systemen nicht vorhanden)
LOGNAME	Aktueller Loginname (auf manchen UNIX-Systemen nicht vorhanden)
UID	Benutzer-ID [user id]
EUID	Effektive Benutzer-ID [effective user id]
HOST	Rechner-Name
HOSTNAME	Rechner-Name
HOSTTYPE	Rechner-Architektur
LANG	Spracheinstellung (C, de_DE.UTF-8, en_EN.iso8859, ...)
LANGUAGE	Spracheinstellung (C, de_DE.UTF-8, en_EN.iso8859, ...)
LC_ALL	Spracheinstellung (C, de_DE.UTF-8, en_EN.iso8859, ...)
GDM_LANG	Spracheinstellung für GUI (C, de_DE.UTF-8, en_EN.iso8859, ...)
SHELL	Name der Login-Shell
SHELLOPTS	Shell-Optionen
SHLVL	Shell-Ebene (Anzahl verschachtelter Shells)
IFS	Trennzeichen für Befehl "read" [internal field separator]

PS1	Shell-Standard-Prompt ("$ ")
PS2	Shell-Fortsetzungs-Prompt ("> ")
PS3	Shell-Prompt für select-Kommando ("#? ")
PS4	Shell-Prompt für Option "-e" (execute) ("+ ")
PAGER	Name des Standard-Pager-Programms (less, more, ...)
EDITOR	Definiert automatisch aufgerufenen Standard-Editor
VISUAL	Definiert automatisch aufgerufenen Standard-Editor
TERM	Terminaltyp (für Editor, more/less, curses-Bibliothek)
COLUMNS	Anzahl Spalten im Terminal
LINES	Anzahl Zeilen im Terminal
LPDEST	Name des Standard-Druckers [line print destination]
PRINTER	Name des Standard-Druckers
DISPLAY	Display-Name für X11-Fenstersystem
TZ	Zeitzone [time zone]
OSTYPE	Betriebssystem-Typ [operating system type]
RANDOM	Zufallszahl (0..32767)
GREPCOLOR	Farbeinstellungen für "grep"-Kommando
LESS	Optionen für Programm "less"
LS_COLORS	Farbeinstellungen für Programm "ls"
LS_OPTIONS	Farbeinstellungen für Programm "ls"

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	Sicherer Wert-Zugriff, falls Text "xxx" direkt dahinter
VAR=	Löschen einer Shell-Variablen (anschließend leer)
unset VAR	Löschen einer Shell-Variablen (anschließend undefiniert)
set	Alle Shell-Variablen (+ Funktionen) auflisten
typeset -	Alle Shell-Variablen (+ Funktionen) auflisten

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

Ein spezieller Typ von Shell-Variablen sind die 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 eines Kindprozesses 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-Variable in Umgebungs-Variable umwandeln
export VAR=TEXT	Analog + gleichzeitige Wertzuweisung (nur "bash")
declare -x VAR	Analog
$VAR	Zugriff auf den Wert (Inhalt) einer Shell-Variablen
${VAR}xxx	Sicherer Wert-Zugriff, falls Text "xxx" direkt dahinter
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
export	Alle Umgebungs-Variablen auflisten
declare -x	Alle Umgebungs-Variablen auflisten

HINWEIS:

* Bei der Zuweisung sind KEINE Leerzeichen um das "=" erlaubt.

* Konvention: In der Shell-Programmierung werden Variablen GROSS geschrieben (wahrscheinlich um sie besser erkennen zu können, da alles andere klein geschrieben wird).

* Shell-Variable sind nur für die Shell selbst relevant, externe Kommandos sehen diese NICHT.

* Umgebungs/Environment-Variable sind für EXTERNE Kommandos relevant, sie werden automatisch an von der Shell aufgerufene externe Kommandos weitergegeben.

* Eine Shell-Variable wird mit "export" in eine Umgebungs-Variable umgewandelt (die umgekehrte Richtung ist nur möglich durch Löschen mit "unset" und Neudefinition der Variable).

* Die Reihenfolge von Zuweisung eines Werts an eine Variable und exportieren der Variable ist egal. In der "bash" sind diese beiden Anweisungen sogar in einer Anweisung kombinierbar.

* Shell-Variable werden auch als "lokale" Variable bezeichnet (lokal zur Shell).

* Umgebungs-Variable werden auch als "globale" Variable bezeichnet. Dies ist irreführend, eine bessere Bezeichnung wäre "vererbte" Variable.

* Umgebungs-Variable werden rekursiv an alle Sub-Prozesse weitervererbt.

* Zur Ausführung von Shell-Skripten (Kommando/Batch-Prozeduren) wird immer eine "Sub-Shell" (d.h. ein Kindprozess) gestartet. Ein Teil der Variablen darin wird vom Elternprozess geerbt. Alle Variablen sind "lokal" zu diesem Prozess und "sterben" zusammen mit ihm, d.h. können keine Wirkung auf den Elternprozess haben

* Aliase und Funktionen werden grundsätzlich nicht an Sub-Shells "vererbt".

* Es gibt KEINE GLOBALEN Variablen in Linux, da Variablen immer Bestandteil eines Prozesses sind. D.h. Variablen "sterben" mit dem Prozess, in dem sie definiert sind.

3) Bedingte Bewertung von Shell-Variablen

Beim Zugriff auf den Wert einer Shell-Variable sind neben den Standardformen "$VAR" bzw. "${VAR}" noch folgende Varianten möglich:

Zugriff	Bedeutung
${VAR-TEXT}	VAR falls DEFINIERT, sonst TEXT
${VAR=TEXT}	Analog + Zuweisung von TEXT an VAR
${VAR+TEXT}	TEXT falls DEFINIERT, sonst nichts
${VAR?}	Ausgabe "VAR: parameter null or not set" + Abbruch falls UNDEFINIERT, sonst VAR
${VAR?TEXT}	Analog mit Ausgabe von "VAR: TEXT"
${VAR:-TEXT}	VAR falls NICHT LEER, sonst TEXT

```
  | ${VAR:=TEXT} |   Analog + Zuweisung von TEXT an VAR               |
  | ${VAR:+TEXT} | TEXT falls NICHT LEER, sonst nichts                |
  | ${VAR:?}     | Ausgabe "VAR: parameter null or not set" + Abbruch |
  |              |   falls VAR LEER, sonst VAR                        |
  | ${VAR:?TEXT} |   Analog mit Ausgabe von "VAR: TEXT"               |
  +--------------+----------------------------------------------------+
```

HINWEIS:

* Ein Doppelpunkt nach VAR verlangt, dass die Variable 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)

Das unterschiedliche Verhalten der Shell- und der Umgebungs-Variablen wird
mit folgendem Shell-Skript demonstriert:

```
  # Übergebene Werte ausgeben
  echo "SHVAR=$SHVAR"
  echo "ENVAR=$ENVAR"

  # Werte neu belegen
  SHVAR="neu_sss"
  ENVAR="neu_eee"

  # Neu belegte Werte ausgeben
  echo "SHVAR=$SHVAR"
  echo "ENVAR=$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   # Variablen-Inhalt 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=neu_sss         # --> Variablen-Inhalt in Sub-Shell
  ENVAR=neu_eee         # --> Variablen-Inhalt in Sub-Shell
                        # Skriptende
  $ echo $SHVAR $ENVAR  # Variablen-Inhalt in Login-Shell ausgeben
  sss eee               # --> Ergebnis
```

5) Variable 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	Shell-Variable erzeugen + Text zuweisen
@ VAR = TEXT	Analog
$VAR	Zugriff auf Wert (Inhalt) einer Shell-Variable
${VAR}xxx	Sicherer Wert-Zugriff, falls Text direkt dahinter
set VAR =	Löschen einer Shell-Variable (anschließend leer)
unset VAR	Löschen einer Shell-Variable (anschließend undefiniert)
set	Alle Shell-Variable auflisten
@	Analog
setenv VAR	Erzeugt eine Umgebungs-Variable
setenv VAR TEXT	Analog + gleichzeitige Wertzuweisung
$VAR	Zugriff auf den Wert (Inhalt) einer Umgebungs-Variable
${VAR}xxx	Sicherer Wert-Zugriff, falls Text direkt dahinter
setenv VAR	Löschen einer Umgebungs-Variable (anschließend leer)
unsetenv VAR	Löschen einer Umgebungs-Variable (anschließend undef.)
env	Alle Umgebungs-Variablen auflisten
printenv	Analog

* 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-Variable ändert sich (meist) auch die Umgebungs-Variable, bei Änderung der Umgebungs-Variable 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 der Shell-Familien "sh" und "csh":

	Shell-Familie	
Typ	sh (bash, ksh, zsh)	csh (tcsh)

	VAR=	set var =	
		@ var =	
	VAR=WERT	set var = WERT	
Shell-Variable		@ var = WERT	
	set	set	
		@	
	echo $VAR	echo $var	
	unset VAR	unset var	
Shell-Array	VAR=(WERT1 WERT2...) (bash!)	set var = (WERT1 WERT2 ...)	
	VAR=([0]=W1 [2]=2...) (bash!)		
	VAR[N]=WERTn (nur bash!)		
	echo ${VAR[N]} (nur bash!)	echo $var[N]	
	echo ${VAR[*]} (nur bash!)		
	echo ${VAR[@]} (nur bash!)		
	echo ${#VAR[*]} (nur bash!)		
Umgebungs-Variable Environ-ment-Var.	export VAR=	setenv VAR	
	export VAR=WERT (nur bash!)	setenv VAR WERT	
		setenv VAR ${WERT1}:${WERT2}:...	
	export VAR		
	env	(print)env	
	echo $VAR	echo $VAR	
	unset 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 per "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": Indizierungmit mit numerischem Index ab 1