HOWTO zu Shell- und Umgebungs/Environment-Variablen

$Id: shell-variable-HOWTO.txt,v 1.18 2019/11/26 19:37:07 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 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 lÃ¶schen). Die wichtigsten Shell-Variablen sind:

Name	Beschreibung
CDPATH	Suchpfad fÃ¼r Kommando "cd"
EDITOR	Definiert den Standard-Editor (vi/emacs)
HOME	Standardverzeichnis fÃ¼r "cd" (Home-Verzeichnis)
IFS	Whitespace-Zeichen fÃ¼r read-Befehl ("internal field separator")
LOGNAME	Aktueller Loginname (auf manchen UNIX-Systemen nicht vorhanden)
MANPATH	Suchpfad fÃ¼r man-Kommando
PATH	Suchpfad fÃ¼r Kommandoaufruf
PS1	Shell-Prompt ("$ ")
PS2	Fortsetzungs-Prompt ("> ")
PWD	Aktuelles Verzeichnis ("print working directory")
RANDOM	Zufallszahl (0..32767)
SHELL	Name der Login-Shell
TERM	Terminaltyp (fÃ¼r 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 LÃ¶schen 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=	LÃ¶schen einer Shell-Variablen (anschlieÃM-^_end leer)
unset VAR	LÃ¶schen einer Shell-Variablen (anschlieÃM-^_end 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 kÃ¶nnen die vererbte Variablen-Liste anschlieÃM-^_end beliebig verÃ¤ndern und erweitern und diese geÃ¤nderte Liste

ihrerseits an eigene Kindprozesse weitervererben. In den Elternprozess "zurÃ¼ckschreiben" kÃ¶nnen 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=	LÃ¶schen einer Umgeb.-Variablen (anschlieÃM-^_end leer)
unset VAR	LÃ¶schen einer Umgeb.-Variablen (anschlieÃM-^_end 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 kÃ¶nnen, da alles andere klein geschrieben wird).

* Shell-Variablen sind nur fÃ¼r die Shell selbst relevant, externe Kommandos sehen sie NICHT.

* Umgebungs/Environment-Variablen sind fÃ¼r 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 mÃ¶glich durch lÃ¶schen mit "unset" und Neudefinition der Variablen).

* Die Reihenfolge von Zuweisung eines Werts zu einer Variablen und exportieren der Variablen ist egal. In der "bash" kÃ¶nnen 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 wÃ¤re "vererbte" Variablen (an alle Subshells einer Shell).

* FÃ¼r die AusfÃ¼hrung von Shell-Skripten (Kommando/Batch-Prozeduren) wird immer eine "Sub-Shell" (d.h. ein Kindprozess) gestartet.

* Aliase und Funktionen werden grundsÃ¤tzlich 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 mÃ¶glich:

Zugriff	Bedeutung
${VAR-TEXT}	VAR zurÃ¼ck, falls VAR definiert; sonst TEXT zurÃ¼ck
${VAR=TEXT}	Analog + Zuweisung von TEXT an VAR
${VAR+TEXT}	TEXT zurÃ¼ck, falls VAR definiert; sonst nichts zurÃ¼ck
${VAR?}	Ausgabe "VAR: parameter null or not set"
${VAR?TEXT}	+ Abbruch falls VAR undefiniert; sonst VAR zurÃ¼ck Analog Ausgabe von "VAR: TEXT"; sonst VAR zurÃ¼ck
${VAR:-TEXT}	VAR zurÃ¼ck, falls VAR leer; sonst TEXT zurÃ¼ck
${VAR:=TEXT}	Analog + Zuweisung von TEXT an VAR
${VAR:+TEXT}	TEXT zurÃ¼ck, falls VAR leer; sonst nichts zurÃ¼ck
${VAR:?}	Ausgabe "VAR: parameter null or not set"
${VAR:?TEXT}	+ Abbruch falls VAR leer; sonst VAR zurÃ¼ck Analog Ausgabe von "VAR: TEXT"; sonst VAR zurÃ¼ck

HINWEIS:

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

4) Beispiele

```
4.1) Variable PATH erweitern
----------------------------

  echo $PATH                    Inhalt von Variable PATH anzeigen
  PATH=                         Variable PATH lÃ¶schen
  PATH="/bin:/usr/bin"          Variable PATH setzen ("..." auch weglassbar)
  PATH="$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 verkÃ¼rzen
----------------------------

  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] # ÃM-^\bergebene 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"

AusfÃ¼hrung 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 LÃ¶schen 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 =       | LÃ¶schen einer Shell-Variablen (anschlieÃM-^_end leer) |
  | unset VAR       | LÃ¶schen einer Shell-Variablen (anschlieÃM-^_end 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	LÃ¶schen einer Umgebungs-Variablen (anschlieÃ-^_end leer)
unsetenv VAR	LÃ¶schen einer Umgebungs-Variablen (anschlieÃ-^_end 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 mÃ¶glich.

* In der "csh" gibt es einige Variablen, die bis auf die GroÃ-^_/Kleinschreibung gleich heiÃ-^_en und auch den gleichen Wert enthalten: HOME/home, PATH/path, TERM/term, PWD/cwd, SHELL/shell, USER/user. Bei Ã-^Dnderung der Shell-Variablen Ã¤ndert sich (meist) auch die Umgebungs-Variable, bei Ã-^Dnderung der Umgebungs-Variablen bleibt die Shell-Variable unverÃ¤ndert:

```
  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":

	Shell-Familie	
Typ	sh (bash, ksh)	csh (tcsh)
Shell-Variable	VAR= VAR=WERT set echo $VAR unset VAR	set var = @ var = set var = WERT @ var = WERT set @ 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!) env echo $VAR unset VAR	setenv VAR setenv VAR WERT setenv VAR ${WERT1}:${WERT2}:... (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, dÃ¼rfen 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 LÃ¶schen 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 mÃ¶glich.

* Array-Variablen sind in der "bash" und in der "csh" mÃ¶glich.
  + "bash": Indizierung mit numerischem Index ab 0 sowie mit Texten (assoziativ/Hash)

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