

# Shell-Praxis (Teil 12)



## Shell-Variablen

# Inhaltsverzeichnis

- [Variablen-Zuweisung](#)
- [Variablen-Substitution](#)
- [Exportieren von Variablen](#)
- [Anzeige aller exportierten Variablen](#)
- [Löschen von Variablen](#)
- [Variablen mit leeren Werten](#)
- [Die wichtigsten automatisch gesetzten Variablen](#)

# Variablen-Zuweisung

Syntax: `NAME=WERT`

- `NAME` ist der Name der Variablen.
- `WERT` ist der Wert der Variablen.
- Erlaubte Zeichen für den Variablennamen sind Buchstaben, Ziffern und Unterstriche. Das erste Zeichen darf keine Ziffer sein.
- Groß- und Kleinschreibung beim Namen sind signifikant (anders als in Windows).
- Vor und nach dem `=` dürfen keine Leerzeichen stehen.

- Enthält der Wert Leerzeichen oder andere Shell-Sonderzeichen, muss der Wert ggf. in Anführungszeichen gesetzt werden.
- Soll innerhalb des Wertes eine Variablen- oder Kommandosubstitution erfolgen, muss der Wert in doppelten Anführungszeichen stehen.
- Eine so gesetzte Variable hat nur in der aktuellen Shell Gültigkeit. Sie ist in Kind-Prozessen nicht verfügbar.

# Variablen-Substitution

- Um eine Variable zu referenzieren (den Wert der Variablen zu verwenden), wird dem Variablennamen ein `$`-Zeichen vorangestellt.
- Die Variablen-Substitution erfolgt auch innerhalb von doppelten Anführungszeichen. Sie unterbleibt innerhalb von einfachen Anführungszeichen. Das `$`-Zeichen ist innerhalb von einfachen Hochkommata vor der Shell geschützt. Auch durch einen vorangestellten Backslash (`\`) kann das `$`-Zeichen geschützt, d.h. die Variablen-Substitution verhindert werden.

```
hermann@debian:~$ PLZ=12345
hermann@debian:~$ echo $PLZ
12345
hermann@debian:~$ echo "My zip code is: $PLZ"
My zip code is: 12345
hermann@debian:~$ myAddress="Badstr. 123, $PLZ Musterstadt"
hermann@debian:~$ echo $myAddress
Hermann Hueck, Bad 123, 12345 Musterstadt
hermann@debian:~$ echo "I live in $myAddress."
I live in Badstr. 123, 12345 Musterstadt.
hermann@debian:~$ cd my-tests
hermann@debian:~/my-tests$ echo "my current working directory is: $(pwd)"
my current working directory is: /home/hermann/my-tests
hermann@debian:~/my-tests$ echo "my current working directory is: $PWD"
my current working directory is: /home/hermann/my-tests
hermann@debian:~/my-tests$ echo "my previous working directory was: $OLDPWD"
my previous working directory was: /home/hermann
```

```
hermann@debian:~/my-tests$ ps
  PID TTY          TIME CMD
 1024 pts/1        00:00:00 bash
 1049 pts/1        00:00:00 ps
hermann@debian:~/my-tests$ bash # start a subshell
hermann@debian:~$ ps
hermann@debian:~/my-tests$ ps
  PID TTY          TIME CMD
 1024 pts/1        00:00:00 bash
 1050 pts/1        00:00:00 bash
 1055 pts/1        00:00:00 ps
```

```
hermann@debian:~/my-tests$ echo $PLZ # $PLZ not available in subshell

hermann@debian:~/my-tests$ echo $myAddress # $myAddress not available in subshell

hermann@debian:~/my-tests$ echo $PWD # $PWD is available in subshell
/home/hermann/my-tests
hermann@debian:~/my-tests$ echo $OLDPWD # $OLDPWD is available in subshell
/home/hermann
hermann@debian:~/my-tests$ exit
hermann@debian:~/my-tests$ ps
  PID TTY          TIME CMD
 1024 pts/1        00:00:00 bash
 1058 pts/1        00:00:00 ps
```

Wie dieses Beispiel zeigt, sind die Variablen `PLZ` und `myAddress` in der Subshell nicht verfügbar. Die Variablen `PWD` und `OLDPWD` sind in der Subshell verfügbar.



# Exportieren von Variablen

Bei der Variablenzuweisung wie oben gezeigt, ist die Variable nur in der aktuellen Shell verfügbar. Soll die Variable auch in Kind-Prozessen verfügbar sein, muss die Variable exportiert werden. Der Variablenzuweisung wird dann das Kommando `export` vorangestellt.

## Syntax:

- `NAME=WERT; export NAME` - zwei Kommandos (können auch durch eine neue Zeile getrennt werden)
- `export NAME=WERT` - Variablenzuweisung und Export in einem

Kommando

```
hermann@debian:~/my-tests$ export PLZ=12345
hermann@debian:~/my-tests$ echo $PLZ
12345
hermann@debian:~/my-tests$ echo "My zip code is:  $PLZ"
My zip code is:  12345
hermann@debian:~/my-tests$ export myAddress="Badstr. 123, $PLZ Musterstadt"
hermann@debian:~/my-tests$ echo $myAddress
Hermann Hueck, Bad 123, 12345 Musterstadt
hermann@debian:~/my-tests$ echo "I live in $myAddress."
I live in Badstr. 123, 12345 Musterstadt.
```

Die folgende Folie zeigt, dass die Variablen `PLZ` und `myAddress` durch das Exportieren auch in der Subshell (allgemeiner: in allen Kind-Prozessen) verfügbar sind.

```
hermann@debian:~/my-tests$ ps
  PID TTY          TIME CMD
 1024 pts/1        00:00:00 bash
 1063 pts/1        00:00:00 ps
hermann@debian:~/my-tests$ bash # start a subshell
hermann@debian:~$ ps
hermann@debian:~/my-tests$ ps
  PID TTY          TIME CMD
 1024 pts/1        00:00:00 bash
 1065 pts/1        00:00:00 bash
 1069 pts/1        00:00:00 ps
hermann@debian:~/my-tests$ echo $PLZ # $PLZ is available in subshell
12345
hermann@debian:~/my-tests$ echo $myAddress # $myAddress is available in subshell
Badstr. 123, 12345 Musterstadt
hermann@debian:~/my-tests$ exit
hermann@debian:~/my-tests$ ps
  PID TTY          TIME CMD
 1024 pts/1        00:00:00 bash
 1072 pts/1        00:00:00 ps
```

# Anzeige aller exportierten Variablen

Das Kommando `export` ohne Argumente zeigt alle exportierten Variablen an.

```
hermann@debian:~/my-tests$ export foo=bar
hermann@debian:~/my-tests$ export
declare -x HOME="/home/hermann"
declare -x LANG="de_DE.UTF-8"
declare -x LOGNAME="hermann"
...
declare -x OLDPWD="/home/hermann"
declare -x PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/home/hermann/bin"
declare -x PWD="/home/hermann/my-tests"
declare -x SHELL="/bin/bash"
declare -x USER="hermann"
declare -x foo="bar"
```

# Löschen von Variablen

Syntax: `unset NAME`

Wird eine Variable mit `unset` gelöscht, dann ist sie weder in der aktuellen Shell noch in Kind-Prozessen verfügbar.

```
hermann@debian:~/my-tests$ export foo=bar
hermann@debian:~/my-tests$ bash # start sub shell
hermann@debian:~/my-tests$ echo $foo
bar
hermann@debian:~/my-tests$
exit
hermann@debian:~/my-tests$ unset foo
hermann@debian:~/my-tests$ bash # start sub shell
hermann@debian:~/my-tests$ echo $foo

hermann@debian:~/my-tests$ exit
exit
hermann@debian:~/my-tests$
```

# Variablen mit leeren Werten

Oftmals wird die Variable nicht gelöscht, sondern sie wird mit einem leeren Wert belegt. Das erfolgt durch eine Zuweisung ohne Wert: `NAME=` oder `NAME=""`. Dies wirkt sich auch auf die Kind-Prozesse aus.

Dies ist technisch etwas anderes als das Löschen der Variablen. Die Variable existiert weiterhin, ihr Wert ist jedoch die leere Zeichenkette.

Die praktische Auswirkung von Löschen oder Leer-Setzen einer Variablen ist in den meisten Fällen gleich. Die kleinen Unterschiede besprechen wir hier nicht.

```
hermann@debian:~/my-tests$ export foo=bar
hermann@debian:~/my-tests$ bash # start sub shell
hermann@debian:~/my-tests$ echo $foo
bar
hermann@debian:~/my-tests$ exit # exit sub shell
exit
hermann@debian:~/my-tests$ foo=""
hermann@debian:~/my-tests$ bash # start sub shell
hermann@debian:~/my-tests$ echo $foo

hermann@debian:~/my-tests$ exit # exit sub shell
exit
hermann@debian:~/my-tests$
```



# Die wichtigsten automatisch gesetzten Variablen

- **USER** - der Benutzername des Benutzers
- **LOGNAME** - der Benutzername des Benutzers
- **HOME** - das Heimat-Verzeichnis des Benutzers
- **SHELL** - der absolute Pfad zur Shell des Benutzers
- **LANG** - die Spracheinstellung des Benutzers

- **PATH** - die durch Doppelpunkte getrennte Liste der Verzeichnisse, in denen die Shell (von links nach rechts) nach ausführbaren Dateien sucht
- **PWD** - das aktuelle Arbeitsverzeichnis (diese Variable wird beim Wechsel des Arbeitsverzeichnisses mit **cd** automatisch aktualisiert)
- **OLDPWD** - das vorherige Arbeitsverzeichnis (diese Variable wird beim Wechsel des Arbeitsverzeichnisses mit **cd** automatisch aktualisiert)