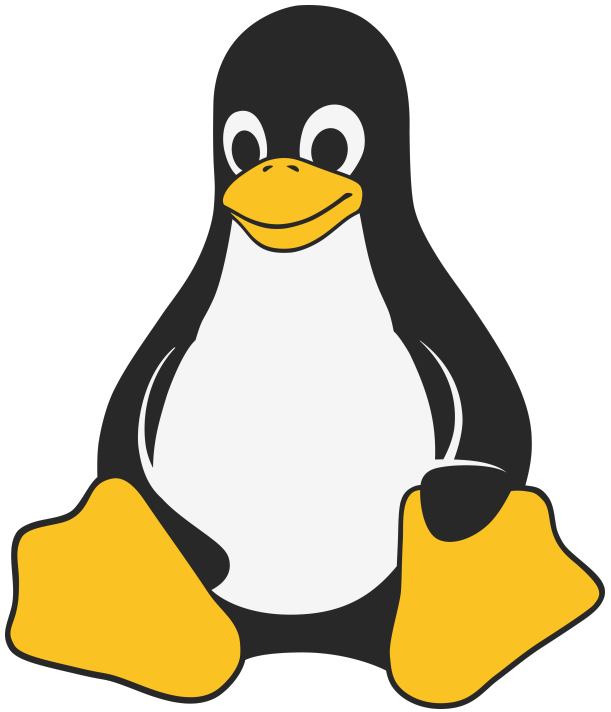
Shell-Variablen



Inhaltsverzeichnis

- Key Takeaways
- Aufgaben

Variablen-Definition

Syntax: `NAME=WERT`

- `NAME` ist der Name der Variablen.
- `WERT` ist der Wert der Variablen.

Regeln:

- 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 (z.B. Subshells) 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 Anführungszeichen vor der Shell geschützt. Auch durch einen vorangestellten Backslash (`\`) kann das `$`-Zeichen geschützt werden, d.h. die Variablen-Substitution wird verhindert.

```
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
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$ echo $$ # PID of the current shell
33333
hermann@debian:~/my-tests$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
hermann    33333   33332  0 14:21 pts/1    00:00:00 -bash
hermann    33432   33333 99 15:22 pts/1    00:00:00 ps -f
```

```
hermann@debian:~/my-tests$ bash # start subshell
hermann@debian:~/my-tests$ echo $$ # PID of the subshell
33434
hermann@debian:~/my-tests$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
hermann    33333   33332  0 14:21 pts/1    00:00:00 -bash
hermann    33434   33333  0 15:23 pts/1    00:00:00 bash
hermann    33438   33434  0 15:23 pts/1    00:00:00 ps -f
```

```
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 $USER # $USER is available in subshell
hermann
hermann@debian:~/my-tests$ echo $HOME # $HOME is available in subshell
/home/hermann
hermann@debian:~/my-tests$ exit # exit subshell
```

```
hermann@debian:~/my-tests$ echo $$
33333
hermann@debian:~/my-tests$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
hermann    33333   33332  0 14:21 pts/1    00:00:00 -bash
hermann    33439   33333  0 15:25 pts/1    00:00:00 ps -f
```

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

`PLZ` und `myAddress` wurden nicht exportiert. `USER` und `HOME` sind (automatisch gesetzte) und exportierte Variablen.

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:

- `export NAME=WERT` - ein Kommando für Zuweisung und Export
- `NAME=WERT; export NAME` - zwei Kommandos für Zuweisung und Export

```
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.
```

```
hermann@debian:~/my-tests$ echo $$ # PID of the current shell
33333
hermann@debian:~/my-tests$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
hermann    33333   33332  0 14:21 pts/1    00:00:00 -bash
hermann    33441   33333  0 15:27 pts/1    00:00:00 ps -f
```

```
hermann@debian:~/my-tests$ bash # start a subshell
hermann@debian:~$ echo $$ # PID of the subshell
33443
hermann@debian:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
hermann    33333   33332  0 14:21 pts/1    00:00:00 -bash
hermann    33443   33333  0 15:28 pts/1    00:00:00 bash
hermann    33447   33443  0 15:29 pts/1    00:00:00 ps -f
```

Durch dem Exportieren sind die Variablen `PLZ` und `myAddress` auch in Subshells (allgemeiner: in allen Kind-Prozessen) verfügbar.

```
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 # exit subshell
```

```
hermann@debian:~/my-tests$ echo $$
33333
hermann@debian:~/my-tests$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
hermann    33333   33332  0 14:21 pts/1    00:00:00 -bash
hermann    33455   33333  0 15:34 pts/1    00:00:00 ps -f
```

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 | nl -ba
     3  declare -x HOME="/home/hermann"
     4  declare -x LANG="de_DE.UTF-8"
     5  declare -x LOGNAME="hermann"
...
     8  declare -x OLDPWD="/home/hermann"
     9  declare -x PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:..."
    10  declare -x PWD="/home/hermann/my-tests"
    11  declare -x SHELL="/bin/bash"
...
    17  declare -x USER="hermann"
...
    22  declare -x foo="bar"
```

Anzeige aller Variablen

Das Kommando `set` zeigt alle Variablen an, die in der aktuellen Shell definiert sind. Es zeigt auch die Shell-Funktionen an, die uns an dieser Stelle nicht interessieren. Die Ausgabe ist oft sehr lang. Mit `set | nl -ba | less` kann die Ausgabe nummeriert und seitenweise geblättert werden.

```
hermann@debian:~/my-tests$ set | nl -ba | head -60
     1  BASH=/bin/bash
...
    23  HOME=/home/hermann
    24  HOSTNAME=debian
...
    27  LANG=de_DE.UTF-8
...
    29  LOGNAME=hermann
...
    37  PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:...
...
    39  PPID=33332
    40  PS1='\[\e]0;\u@\h: \w\a\]...\$ '
    41  PS2='> '
...
    43  PWD=/home/hermann
    44  SHELL=/bin/bash
...
    51  UID=1000
    52  USER=hermann
...
    56  XDG_SESSION_TYPE=tty
```

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.

Zum Inhaltsverzeichnis ...

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

```
hermann@debian:~/my-tests$ unset foo
hermann@debian:~/my-tests$ echo $foo # foo not available in current shell

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

hermann@debian:~/my-tests$ exit # exit subshell
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.

Zum Inhaltsverzeichnis ...

```
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$
```

Automatisch gesetzte 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
- `PS1` - der (primäre) Shell-Prompt des Benutzers
- `PS2` - der sekundäre Prompt (bei mehrzeiligen Kommando-Eingaben)

Zum Inhaltsverzeichnis ...

- `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)
- `$?` - der Endestatus/Rückgabewert des zuletzt ausgeführten Kommandos (wird nach jeder Ausführung eines Kommandos automatisch gesetzt)

- `$$` - die Prozess-ID der Shell (wird beim Start einer Shell automatisch gesetzt)
- `$PPID` - die Prozess-ID des Eltern-Prozesses der Shell (wird beim Start einer Shell automatisch gesetzt)
- Es gibt weitere. Die hier genannten sind die wichtigsten.
- Positionsparameter (`$1`, `$2`, ... `$9`, `$0`, `$#`, `$*`, `$@`) sind ebenfalls Variablen. Sie spielen vor allem bei Shell-Skripten eine Rolle. Sie werden beim Aufruf eines Shell-Skripts gesetzt. So kann das Skript auf seine Aufrufargumente zugreifen.

Key Takeaways

- Variablen werden mit `NAME=WERT` definiert. So gesetzte Variablen sind nur in der aktuellen Shell verfügbar.
- Um eine Variable in Subshells verfügbar zu machen, wird sie mit dem `export`-Kommando exportiert: `export NAME=WERT` oder `NAME=WERT; export NAME`.
- Um eine Variable zu referenzieren (zu verwenden), wird dem Variablennamen ein `$`-Zeichen vorangestellt. (Variablen-Substitution)

- Die Variablen-Substitution wird verhindert:
  - innerhalb von einfachen Anführungszeichen (`'...'`)
  - durch einen vorangestellten Backslash (`\`)
- Die Variablen-Substitution erfolgt jedoch:
  - innerhalb von doppelten Anführungszeichen (`"..."`)
- `export` ohne Argumente zeigt alle exportierten Variablen an.
- `set` zeigt alle Variablen der aktuellen Shell an.
- `unset NAME` löscht eine Variable.

Zum Inhaltsverzeichnis ...

Aufgaben

- Prüfen Sie in den folgenden Beispielen mit dem Kommando `ps -f`, in welcher Shell Sie sich befinden - in der ersten Terminal-Shell oder in einer Subshell.
- Definieren Sie die Variable `myVar` mit dem Wert `Hello, $USER!`.
- Geben Sie den Wert der Variablen `myVar` aus.
- Starten Sie eine Subshell und geben Sie den Wert der Variablen `myVar` nochmals aus. Ist die Variable in der Subshell verfügbar?
- Verlassen Sie die Subshell.

- Exportieren Sie die Variable `myVar`.
- Geben Sie den Wert der Variablen `myVar` zunächst in der aktuellen Shell aus.
- Starten Sie eine Subshell und geben Sie den Wert der Variablen `myVar` nochmals aus. Ist die Variable in der Subshell verfügbar?
- Verlassen Sie die Subshell wieder.
- Löschen Sie die Variable `myVar`.
- Geben Sie den Wert der Variablen `myVar` aus. Ist die Variable noch verfügbar? Was wird ausgegeben?

- Prüfen Sie, ob die Variable `myVar` in der Subshell noch verfügbar ist.
- Machen Sie dasselbe Experiment, indem Sie die Variable `myVar` mit einem leeren Wert belegen: `myVar=""`. Gibt es offensichtliche Unterschiede zum Löschen der Variablen?
- Lassen Sie sich mit dem Kommando `set | nl -ba | less` alle Variablen mit Zeilennummern anzeigen, die in der aktuellen Shell definiert sind. (`set` gibt außer den Umgebungsvariablen auch die Funktionen aus. Ignorieren Sie die Funktionen.)
- Lassen Sie sich mit dem Kommando `export | nl -ba` alle exportierten Variablen mit Zeilennummern anzeigen.