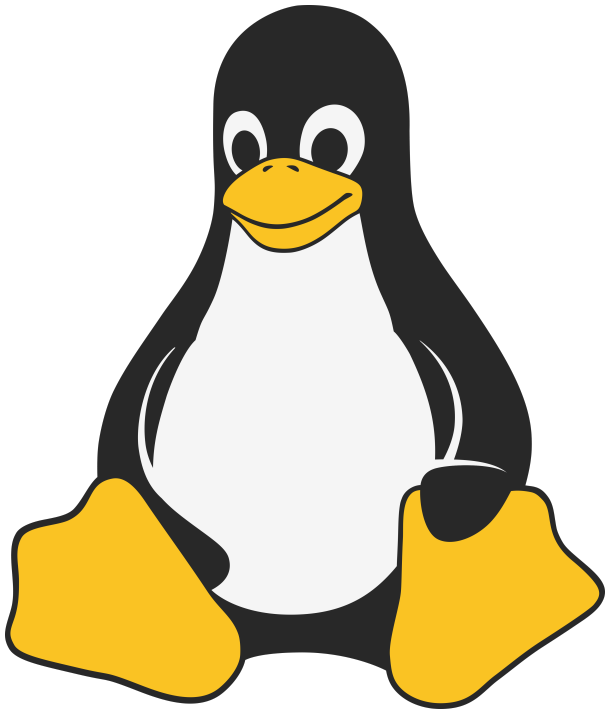
Shell-Variablen (1)



Inhaltsverzeichnis

Beschreibung

- Variablen sind Platzhalter für Werte.
- Man kann selbst Variablen definieren, d.h. einen Namen vergeben und einen Wert zuweisen.
- Bei der Verwendung von Variablen (Variablen-Substitution) wird der Wert der Variablen von der Shell an die Stelle des Variablennamens eingesetzt.

Variablen-Definition

Definition der Variablen `foo` mit dem Wert `bar` (In diesem einfachen Fall ist die Verwendung von Anführungszeichen optional):

```
hermann@debian:~$ foo=bar
```

```
hermann@debian:~$ foo="bar"
```

```
hermann@debian:~$ foo='bar'
```

Kommen Leerzeichen (oder andere Shell-Sonderzeichen) im Variablenwert vor, dann müssen Anführungszeichen verwendet werden:

- Im folgenden Beispiel sind die drei Leerzeichen zwischen `Hallo` und `Welt` Teil des Variablenwerts. Die ganze Zeichenkette ist in doppelten Anführungszeichen vor der Shell geschützt. Deshalb sind die Leerzeichen hier keine Worttrenner, sondern Teil des Variablenwerts.

```
hermann@debian:~$ hello="Hallo   Welt"
```

Regeln:

- Der Variablenname darf nur aus Buchstaben, Ziffern und Unterstrichen bestehen.
- Der Variablenname darf nicht mit einer Ziffer beginnen.
- Vor und nach dem Gleichheitszeichen (`=`) dürfen keine Leerzeichen stehen.
- Bei einfachen Werten darf der Variablenwert in Anführungszeichen stehen, muss aber nicht.
- Enthält der Variablenwert Leerzeichen (oder andere Shell-Sonderzeichen), dann muss er in Anführungszeichen stehen.

Variablen-Substitution

```
hermann@debian:~$ echo $foo
bar
```

```
hermann@debian:~$ echo $hello
Hallo Welt
```

```
hermann@debian:~$ echo "Der Wert von foo ist: $foo"
Der Wert von foo ist: bar
```

```
hermann@debian:~$ echo 'Der Wert von foo ist: $foo'
Der Wert von foo ist: $foo
```

- Um den Wert einer Variablen zu verwenden (d.h. den Variablennamen durch den Wert zu ersetzen), muss dem Variablennamen ein Dollarzeichen (`$`) vorangestellt werden.
- Folgt dem `$`-Zeichen kein gültiger Variablenname (bestehend aus Buchstaben, Ziffern und Unterstrichen), dann wird das `$`-Zeichen als normales Zeichen behandelt.
- Folgt dem `$`-Zeichen ein gültiger Variablenname, zu dem keine Variable definiert ist, dann expandiert die Shell die Variable zu einem Leerstring.

- Um den Wert einer Variablen innerhalb einer Zeichenkette zu verwenden, muss die Zeichenkette in doppelte Anführungszeichen gesetzt werden.
- Dies funktioniert nur mit doppelten Anführungszeichen, nicht mit einfachen Anführungszeichen.

MERKE: **Variablen-Substitution** (und Kommando-Substitution) ist **nur innerhalb von doppelten Anführungszeichen** möglich.

- Der Variablenwert enthält 3 Leerzeichen zwischen `Hallo` und `Welt`.

```
hermann@debian:~$ hello="Hallo   Welt"
```

- Nach der Variablensubstitution parst die Shell den substituierten Wert erneut und findet dann 2 Argumente, die sie an `echo` übergibt. `echo` gibt die Argumente durch ein Leerzeichen getrennt aus.

```
hermann@debian:~$ echo $hello # 3 blanks lost
Hallo Welt
```

- Durch die Anführungszeichen wird der Wert von `"$hello"` als ein Argument (das 3 Leerzeichen enthält) an `echo` übergeben.

```
hermann@debian:~$ echo "$hello" # 3 blanks preserved
Hallo   Welt
```

- Die Ersetzung des Kommandos `echo` durch `ls` zeigt in der Anzahl der Fehlermeldungen, wie viele Argumente an `ls` übergeben wurden.

```
hermann@debian:~$ ls $hello # 2 arguments passed to 'ls'
ls: cannot access 'Hallo': No such file or directory
ls: cannot access 'Welt': No such file or directory
```

```
hermann@debian:~$ ls "$hello" # 1 argument passed to 'ls'
ls: cannot access 'Hallo   Welt': No such file or directory
```

Standard-Variablen

Es gibt einige Standard-Variablen, die von der Shell automatisch belegt werden.

Die meisten dieser Standard-Variablen werden beim Login des Benutzers gesetzt. `$PWD` wird bei jedem Verzeichniswechsel mit `cd` neu gesetzt.

Die wichtigsten Standard-Variablen sind:

Variable	Wert
`$USER`	Benutzername
`$HOME`	Pfad zum Heimat-Verzeichnis des Benutzers
`$PWD`	Pfad zum aktuellen Verzeichnis (gleiche Ausgabe wie das Kommando `pwd`)
`$OLDPWD`	Pfad zum zuvor besuchten Verzeichnis
`$PATH`	durch Doppelpunkte getrennte Verzeichnis-Pfade, in denen die Shell nach ausführbaren Dateien (Programmen oder Skripten) sucht.

$USER und $HOME

```
hermann@debian:~$ echo $USER
hermann
hermann@debian:~$ echo Ich bin $USER.
Ich bin hermann.
hermann@debian:~$ echo $HOME
/home/hermann
hermann@debian:~$ echo Hier wohne ich: $HOME
Hier wohne ich: /home/hermann
hermann@debian:~$ echo "Hier wohnt $USER:     $HOME"
Hier wohnt hermann:     /home/hermann
```

$PWD und $OLDPWD

```
hermann@debian:~$ cd Dokumente
hermann@debian:~/Dokumente$ pwd
/home/hermann/Dokumente
hermann@debian:~/Dokumente$ echo "previous workdir: $OLDPWD, current workdir: $PWD"
previous workdir: /home/hermann, current workdir: /home/hermann/Dokumente
```

```
hermann@debian:~/Dokumente$ cd ..
hermann@debian:~$ pwd
/home/hermann
hermann@debian:~$ echo "previous workdir: $OLDPWD, current workdir: $PWD"
previous workdir: /home/hermann/Dokumente, current workdir: /home/hermann
```

$PATH

```
hermann@debian:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
```

Kürzel `~` für `$HOME`

- Das Kürzel `~` steht für das Heimat-Verzeichnis des Benutzers.
- Es wird auch im Prompt verwendet.
- Es kann in der Shell häufig anstelle von `$HOME` verwendet werden.

```
hermann@debian:~$ echo ~
/home/hermann
```

```
hermann@debian:~$ echo Hier wohne ich: $HOME
Hier wohne ich: /home/hermann
hermann@debian:~$ echo Hier wohne ich: ~
Hier wohne ich: /home/hermann
```

- Anders als `$HOME` wird das Kürzel `~` nicht in doppelten Anführungszeichen substituiert.

```
hermann@debian:~$ echo "Hier wohne ich: $HOME"
Hier wohne ich: /home/hermann
hermann@debian:~$ echo "Hier wohne ich: ~"
Hier wohne ich: ~
```