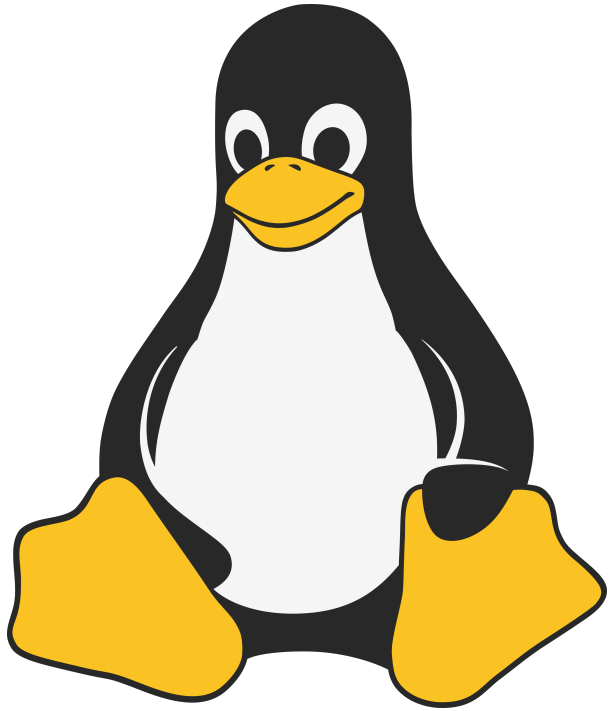
Shell-Praxis

Shell-Sonderzeichen (1)



Inhaltsverzeichnis

Kommando-Trenner

- `\n` (Newline) und `;` sind Kommando-Trenner.

Das Newline-Zeichen (`\n`) verwenden wir schon seit der ersten Kommandoeingabe. Wir schließen ein Shell-Kommando mit der Eingabe-Taste ab. Die Eingabe-Taste erzeugt ein Newline-Zeichen, das die Shell als Kommando-Trenner interpretiert.

Die Shell erkennnt nun das Ende der Kommandozeile, intrtpretiert die eingegebene Zeichenfolge bis zum Newline-Zeichen als Kommando und führt es aus.

Nach Ausführung des Kommandos gibt die Shell den Prompt aus und wartet auf die nächste Eingabe.

Der Semikolon (`;`) ist ein weiterer (semantisch gleichwertiger) Kommando-Trenner. Allerdings gibt die Shell nach der Ausführung des ersten Kommandos keinen neuen Prompt aus, sondern führt sofort das nächste Kommando aus.

```
hermann@debian:~$ cd Dokumente; pwd
/home/hermann/Dokumente
```

Kommentar (#)

Das Hash-Symbol (#) ist das Kommentarzeichen der Shell. Alles, was rechts des Hash-Symbols steht, wird von der Shell ignoriert. (D.h. es wird als Text betrachtet, der für die Kommandoausführung nicht relevant ist.)

```
hermann@debian:~/Dokumente$ pwd # Wo bin ich denn gerade?
/home/hermann/Dokumente
hermann@debian:~/Dokumente$ # Ich geh jetzt gleich nach Hause.
hermann@debian:~/Dokumente$ cd
hermann@debian:~$ pwd # Wo bin ich jetzt?
/home/hermann
```

Bisher behandelte Sonderzeichen

Sonderzeichen	Bedeutung
`\n`	Kommando-Trenner
`;`	Kommando-Trenner
Leerzeichen	Wort-Trenner
`#`	Kommentarzeichen
`$`	Variablen-Substitution
`$(...)`	Kommando-Substitution (wird später behandelt)

Es gibt viele weitere Sonderzeichen.

Maskierung von Sonderzeichen

Manchmal möchte man ein Sonderzeichen seiner besonderen Bedeutung berauben, d.h. das Sonderzeichen soll als normales Zeichen behandelt werden. Z.B. soll das Leerzeichen kein Wort-Trenner sein, sondern ein Leerzeichen. Oder das Semikolon (`;`) soll nicht als Kommando-Trenner, sondern als normales `;` -Zeichen behandelt werden. Oder das Dollarzeichen (`$`) soll nicht als Variablen-Substitution, sondern als `$` -Zeichen (z.B. für eine Preisangabe) behandelt werden.

Soll ein Shell-Sonderzeichen nicht als Sonderzeichen gewertet werden, sondern seine normale Zeichenbedeutung zurückerhalten, so muss es "maskiert" (vor der Shell verstecket) werden.

Die Shell hat eigene Sonderzeichen für die Maskierung von Sonderzeichen.

Sonderzeichen zur Maskierung

Zeichen	Beschreibung	Bedeutung
`'...'`	Einfache Anführungs-zeichen	maskieren alle Zeichen zwischen den einfachen Anführungszeichen ausser `'`
`"..."`	Doppelte Anführungs-zeichen	maskieren alle Zeichen zwischen den doppelten Anführungszeichen ausser `"`, `\` und `$`
`\`	Backslash	maskiert nur das dem Backslash folgende Zeichen

- Die Variablen-Substitution (und Kommando-Substitution) ist in doppelten Anführungszeichen für die Shell sichtbar und wird interpretiert.

Beispiele

```
hermann@debian:~/$ echo Mein Heimatdorf ist     $HOME.
Mein Heimatdorf ist /home/hermann.
hermann@debian:~/$ echo "Mein Heimatdorf ist     $HOME."
Mein Heimatdorf ist     /home/hermann.
hermann@debian:~/$ echo "Mein Heimatdorf ist     \$HOME."
Mein Heimatdorf ist     $HOME.
hermann@debian:~/$ echo 'Mein Heimatdorf ist     $HOME.'
Mein Heimatdorf ist     $HOME.
```

```
hermann@debian:~$ echo 'Der Preis beträgt: $ 100'
Der Preis beträgt: $ 100
hermann@debian:~$ echo "Der Preis beträgt: \$ 100"
Der Preis beträgt: $ 100
hermann@debian:~$ echo Der Preis beträgt: \$ 100
Der Preis beträgt: $ 100
```

```
hermann@debian:~$ echo 'Zitat: "Irren ist menschlich."'
Zitat: "Irren ist menschlich."
hermann@debian:~$ echo "Zitat: \"Irren ist menschlich.\""
Zitat: "Irren ist menschlich."
hermann@debian:~$ echo Zitat: \"Irren ist menschlich.\"
Zitat: "Irren ist menschlich."
```