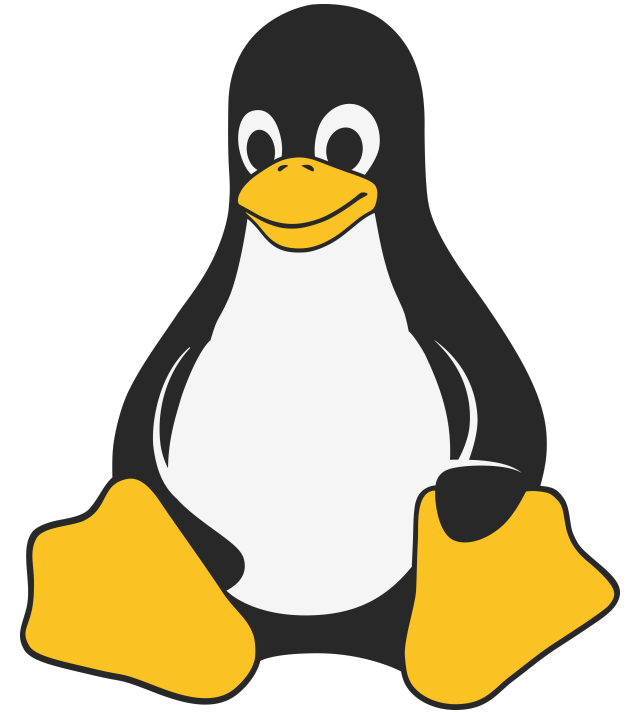


# Shell-Sonderzeichen (2) und deren Maskierung



# Inhaltsverzeichnis

- [Sonderzeichen](#)
- [Maskierung von Sonderzeichen](#)
- [Sonderzeichen zur Maskierung](#)

# Sonderzeichen (Meta-Zeichen)

- Die Shell verwendet eine Reihe von Sonderzeichen, die eine besondere Bedeutung haben.
- Diese Sonderzeichen verlieren (falls sie nicht maskiert sind) ihre normale Zeichenbedeutung und werden von der Shell bei der Verarbeitung der Kommandos (auch in Shell-Skripten) interpretiert.
- Sollen sie ihre Shell-Sonderbedeutung verlieren und ihre Zeichenbedeutung wiedererlangen, müssen sie maskiert werden.

Sonderzeichen	Bedeutung
#	Kommentarzeichen
Leerzeichen	Wort-Trenner, Token-Trenner
>	Ausgabe-Umlenkung
>>	Ausgabe-Umlenkung (Anhängen)
<	Eingabe-Umlenkung
	Pipe und Kommando-Trenner

Sonderzeichen	Bedeutung
<code>\n</code> (Newline)	Kommando-Trenner
<code>;</code>	Kommando-Trenner
<code>&amp;&amp;</code>	Logisches UND und Kommado-Trenner
<code>  </code>	Logisches ODER und Kommando-Trenner
<code>\$</code>	Variablen-Substitution
<code>\$(command)</code>	Kommando-Substitution
<code>`command`</code>	Kommando-Substitution (alte Syntax)

Sonderzeichen	Bedeutung
<code>*</code>	Dateinamens-Wildcard: Platzhalter für eine beliebige Zeichenkette
<code>?</code>	Dateinamens-Wildcard: Platzhalter für ein einzelnes Zeichen
<code>[...]</code>	Dateinamens-Wildcard: Platzhalter für ein einzelnes Zeichen aus einer Zeichenklasse
<code>{cmd1 ; cmd2 ; ...}</code>	Kommandogruppe, Ausführung in der aktuellen Shell
<code>(cmd1 ; cmd2 ; ...)</code>	Kommandogruppe, Ausführung in einer Subshell

Sonderzeichen	Bedeutung
<code>\</code>	Backslash, maskiert das direkt nachfolgende Zeichen
<code>' ... '</code>	Einfache Anführungszeichen, maskieren die eingeschlossenen Zeichen ausser <code>'</code> und <code>\</code>
<code>" ... "</code>	Doppelte Anführungszeichen, maskieren die eingeschlossenen Zeichen ausser <code>"</code> , <code>\</code> , <code>\$</code> und <code>`</code>

- Variablen-Substitution und Kommando-Substitution (auch die alte Syntax mit ``...``) sind in doppelten Anführungszeichen für die Shell sichtbar und werden interpretiert.

# 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 Semikolon ( `;` ) nicht als Kommando-Trenner, sondern als normales Semikolon behandelt werden. Oder das Dollarzeichen ( `$` ) soll nicht als Variablen-Substitution, sondern als Dollarzeichen (z.B. für eine Preisangabe) behandelt werden. In diesen Fällen muss das Sonderzeichen maskiert werden.



- Die Shell hat eigene Sonderzeichen für die Maskierung von Sonderzeichen (s.o.).
- Nach der Bearbeitung der Sonderzeichen durch die Shell werden die Maskierungszeichen ( `\`, `'...'`, `"..."` ) entfernt.

```
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."
```

```
hermann@debian:~$ echo '--'\''--'
--'--
hermann@debian:~$ echo "--'--"
--'--
hermann@debian:~$ echo '--'""'--'
--'--
```