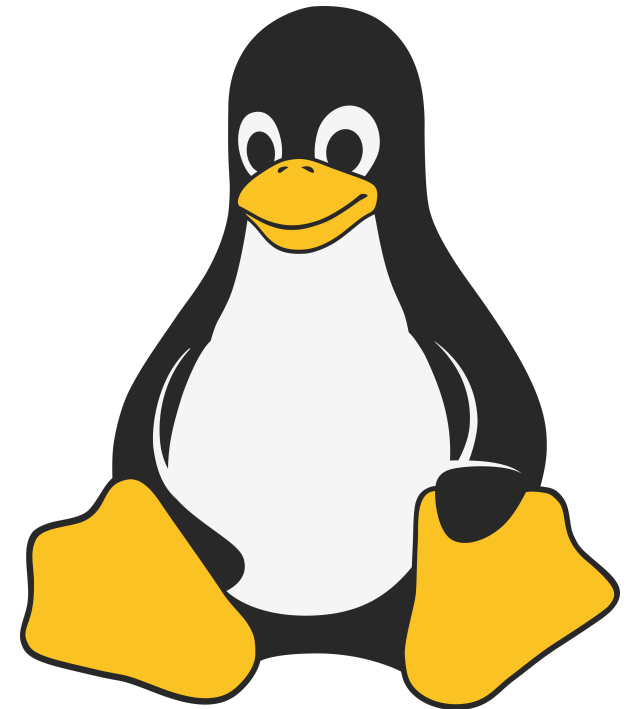


# Shell-Praxis

## Kommando-Substitution



# Kommandosubstitution mit `$(command)`

- Mit `$(command)` wird das Kommando `command` ausgeführt und die Ausgabe des Kommandos an der Stelle der Kommandosubstitution `$(command)` eingesetzt.
- Bei `cowsay $(date)` wird das Kommando `date` ausgeführt und die Ausgabe von `date` an die Stelle von `$(date)` eingesetzt.
- `$(...)` ist ein weiteres Shell-Sonderzeichen.

```
hermann@debian:~$ cowsay $(date)
```

```
-----  
< So 3. Nov 23:39:16 CET 2024 >  
-----
```

```
  \      ^__^  
   \    (oo)\_____  
      (__)\\       )\\/\  
           ||----w |  
           ||     ||
```

Abgesehen von der etwas unterschiedlichen Datumsausgabe, ist die Ausgabe von `cowsay $(date)` und `date | cowsay` gleich.

Die technischen Mechanismen (Kommando-Substitution vs. Pipe-Verknüpfung) sind jedoch unterschiedlich.

```
hermann@debian:~$ date | cowsay
```

```
-----  
< So 3. Nov 23:40:35 CET 2024 >  
-----
```

```
      ^__^  
      (oo)\_____  
      (--)\\       )\\/\  
           ||----w |  
           ||     ||
```

## Kommandosubstitution: alte Syntax : ``command``

- In der alten Syntax der Kommandosubstitution wird das zur Ausführung vorgesehene Kommando in Backticks ``command`` eingeschlossen.
- Dies funktioniert genauso wie die neue Syntax `$(command)`, hat aber Nachteile:
  - Lesbarkeit
  - Verschachtelte Kommandosubstitutionen ist nicht möglich.
- Mit der neuen Syntax können verschachtelte Kommandosubstitutionen durchgeführt werden:  
`$(command1 $(command2))`.

```
hermann@debian:~$ cowsay `date`
```

```
-----  
< So 3. Nov 23:42:37 CET 2024 >  
-----
```

```
      ^__^  
      (oo)\_____  
      (--)\\       )\\/\  
           ||----w |  
           ||     ||
```

```
hermann@debian:~$ cowsay $(echo Heute ist $(date +%A).)
```

```
-----  
< Heute ist Sonntag. >
```

```
-----
```

```
  \      ^__^  
   \    (oo)\_____  
      (__)\\       )\\/\  
         ||----w |  
         ||     ||
```