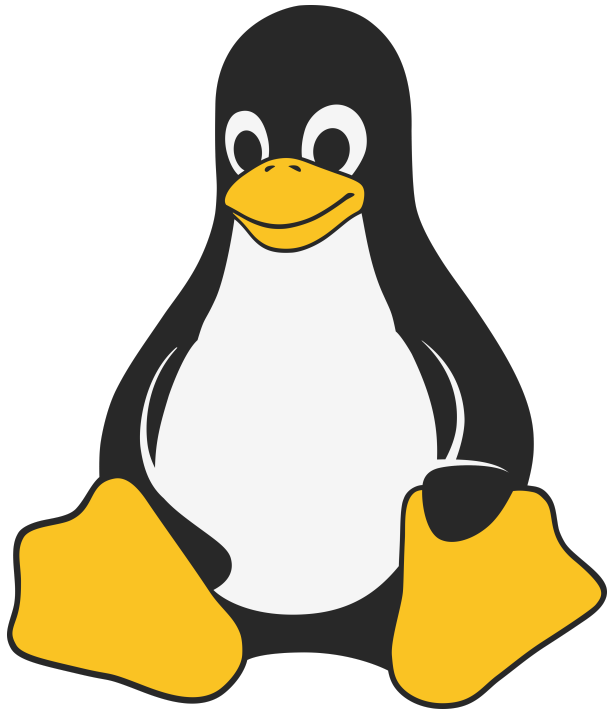
Endestatus (Exit-Code)



Inhaltsverzeichnis

Endestatus (Rückgabewert)

- Jedes Kommando beendet seine Ausführung mit einem Endestatus (oder Rückgabewert).
- `$?` ist eine Umgebungsvariable, die den Endestatus des zuletzt ausgeführten Kommandos enthält. Diese Variable wird nach jedem Kommando automatisch neu gesetzt.
- Der Endestatus ist eine Ganzzahl im Bereich von 0 bis 255.
- `$? = 0`: Die Kommando-Ausführung war erfolgreich.
- `$? != 0`: Die Kommando-Ausführung war nicht erfolgreich. (Die genaue Bedeutung des Endestatus ist kommandospezifisch.)

Zum Inhaltsverzeichnis ...

```
hermann@debian:~$ sudo apt update
hermann@debian:~$ echo $?
0
```

```
hermann@debian:~$ ls /etc/passwd
/etc/passwd
hermann@debian:~$ echo $?
0
```

```
hermann@debian:~$ ls /etc/passwdXXX
ls: Zugriff auf '/etc/passwdXXX' nicht möglich: Datei oder Verzeichnis nicht gefunden
hermann@debian:~$ echo $?
2
```

Logische Kommando-Verknüpfung: `&&`, `||`

- Diese Operatoren verknüpfen zwei Kommandos. Sie werten den Endestatus des linken Kommandos aus und entscheiden, ob das rechte Kommando ausgeführt wird oder nicht.
- `&&` (logisches UND): Das rechte Kommando wird nur ausgeführt, wenn das linke Kommando erfolgreich war.
- `||` (logisches ODER): Das rechte Kommando wird nur ausgeführt, wenn das linke Kommando nicht erfolgreich war.
- Beide Operatoren gelten als Kommando-Trenner.

Zum Inhaltsverzeichnis ...

Beispiele

```
hermann@debian:~$ sudo apt update && sudo apt upgrade
...
```

```
hermann@debian:~$ ls /etc/passwd && echo "Datei /etc/passwd gefunden."
/etc/passwd
Datei /etc/passwd gefunden.
```

```
hermann@debian:~$ ls /etc/passwdXXX || echo "Datei /etc/passwdXXX nicht gefunden."
ls: Zugriff auf '/etc/passwdXXX' nicht möglich: Datei oder Verzeichnis nicht gefunden
Datei /etc/passwdXXX nicht gefunden.
```


Zum Inhaltsverzeichnis ...

Man kann auch drei Kommandos verknüpfen.- zuerst mit `&&` und dann mit `||`.

```
hermann@debian:~$ file=/etc/passwd
hermann@debian:~$ ls $file && echo "Datei $file gefunden." || echo "Datei $file nicht gefunden"
/etc/passwd
Datei /etc/passwd gefunden.
```

```
hermann@debian:~$ file=/etc/passwdXXX
hermann@debian:~$ ls $file && echo "Datei $file gefunden." || echo "Datei $file nicht gefunden"
ls: Zugriff auf '/etc/passwdXXX' nicht möglich: Datei oder Verzeichnis nicht gefunden
Datei /etc/passwdXXX nicht gefunden
```


Zum Inhaltsverzeichnis ...

Oft will man die Ausgabe und Fehlerausgabe des betreffenden Kommandos gar nicht sehen. Man kann sie nach `/dev/null` verwerfen.

```
hermann@debian:~$ file=/etc/passwd
hermann@debian:~$ ls $file > /dev/null 2>&1 && echo "Datei $file gefunden." || echo "Datei $file nicht gefunden"
Datei /etc/passwd gefunden.
```

```
hermann@debian:~$ file=/etc/passwdXXX
hermann@debian:~$ ls $file > /dev/null 2>&1 && echo "Datei $file gefunden." || echo "Datei $file nicht gefunden"
Datei /etc/passwdXXX nicht gefunden
```


Zum Inhaltsverzeichnis ...

Nach den Operatoren `&&` und `||` (auch nach dem Pipe-Operator `|`) kann die physische Kommandozeile umgebrochen werden. (Die Shell erkennt, dass das logische Kommando noch nicht abgeschlossen sein kann.)

```
hermann@debian:~$ file=/etc/passwdXXX
hermann@debian:~$ ls $file &&
> echo "Datei $file gefunden." ||
> echo "Datei $file nicht gefunden"
Datei /etc/passwdXXX nicht gefunden
```

```
hermann@debian:~$ sudo apt update &&
> apt list --upgradable &&
> sudo apt upgrade
...
```

Beispiele mit `grep`

Beim Kommando `grep` wird gerne (vor allem in Skripten) der Endestatus ausgewertet. Die Ausgabe und Fehlerausgabe kann nach `/dev/null` umgeleitet werden. Sie kann auch durch die Optionen `-q` und `-s` unterdrückt werden:

- `grep -q`: (quiet) unterdrückt die Standardausgabe
- `grep -s`: (silent) unterdrückt die Standardfehlerausgabe

Zum Inhaltsverzeichnis ...

```
hermann@debian:~$ file=/etc/passwdXXX pattern=wumpel
hermann@debian:~$ grep $pattern $file
grep: /etc/passwdXXX: Datei oder Verzeichnis nicht gefunden
hermann@debian:~$ echo $?
2
hermann@debian:~$ grep -qs $pattern $file
hermann@debian:~$ echo $?
2
```

```
hermann@debian:~$ file=/etc/passwd pattern=wumpel
hermann@debian:~$ grep $pattern $file
hermann@debian:~$ echo $?
1
hermann@debian:~$ grep -qs $pattern $file
hermann@debian:~$ echo $?
1
```

```
hermann@debian:~$ file=/etc/passwd pattern=hermann
hermann@debian:~$ grep $pattern $file
hermann:x:1000:1000:Hermann Hueck,,,:/home/hermann:/bin/bash
hermann@debian:~$ echo $?
0
hermann@debian:~$ grep -qs $pattern $file
hermann@debian:~$ echo $?
0
```

```
hermann@debian:~$ user=wumpel
hermann@debian:~$ grep -qs $user /etc/passwd &&
> echo "Benutzer $user ist im System registriert." ||
> echo "Benutzer $user ist nicht im System registriert."
Benutzer wumpel ist nicht im System registriert.
```

```
hermann@debian:~$ user=hermann
hermann@debian:~$ grep -qs $user /etc/passwd &&
> echo "Benutzer $user ist im System registriert." ||
> echo "Benutzer $user ist nicht im System registriert."
Benutzer hermann ist im System registriert.
```


Zum Inhaltsverzeichnis ...

`if-then-else`-Konstruktion

```
if [ Kommando ]
then
    Kommandoliste
else
    Kommandoliste
fi
```

- Diese Konstruktion ist semantisch äquivalent zu den logischen Operatoren `&&` und `||`.
- Sie ist insbesondere dann praktisch, wenn im `then`-Block oder im `else`-Block mehrere Kommandos auszuführen sind.

Beispiele mit `if-then-else`

```
hermann@debian:~$ file=/etc/passwd
hermann@debian:~$ if ls $file > /dev/null 2>&1
then
    echo # print empty line
    echo "Datei $file gefunden."
    lines=$(cat $file | wc -l)
    echo "Datei $file enthält $lines Zeilen."
else
    echo "Datei $file nicht gefunden."
fi

Datei /etc/passwd gefunden.
Datei /etc/passwd enthält 37 Zeilen.
```


Zum Inhaltsverzeichnis ...

```
hermann@debian:~$ if sudo apt update
> then
>     echo
>     apt list --upgradable
>     echo
>     sudo apt upgrade
> fi
...
```

```
hermann@debian:~$ user=hermann
hermann@debian:~$ if grep -qs $user /etc/passwd
> then echo "Benutzer $user ist im System registriert."
> else echo "Benutzer $user ist nicht im System registriert."
> fi
Benutzer hermann ist im System registriert.
```

Einfache Kommando-Trenner (Newline, `;`)

Jeder Zeilenumbruch (Newline), der als Kommando-Trenner dient, kann durch ein Semikolon (`;`) ersetzt werden. Beide sind äquivalente Kommando-Trenner.

```
hermann@debian:~$ if sudo apt update; then echo; apt list --upgradable; echo; sudo apt upgrade; fi
...
```


Zum Inhaltsverzeichnis ...