2.4 Skripte/Verzweigung

2.4.1 Was tun bei einem Fehler?

Die bisher erstellten Skripte haben einen Mangel: Wenn ein Programm nicht wie erwartet funktioniert, wird trotzdem der Rest des Skriptes so abgearbeitet, als ob alles in Ordnung wäre. Es fehlt die Möglichkeit, auf solch einen Fall passend zu reagieren.

Diese Möglichkeit bietet uns die *Verzweigung*. In fast allen Programmiersprachen ist sie vorhanden, und meistens wird sie dort mit dem Schlüsselwort `if` eingeleitet. So ist es auch hier.

2.4.2 Beispiel: Ein Backup-Skript

Das folgende Skript soll das Verzeichnis `daten` in die Datei `daten.tar` sichern. Anschließend wird die Datei `daten.tar` mit dem Programm `gzip` komprimiert (das geht auch einfacher).

simplebackup1.sh

```
1 #!/bin/bash
2 tar -cvf ~/daten.tar ~/daten/
3 gzip ~/daten.tar
```

Der zweite Schritt soll aber nur getan werden, wenn der erste geklappt hat. Aber woher weiß man, ob der erste Befehl gelungen ist? Dazu gibt es den Rückgabewert von Prozessen. Jeder Prozess gibt einen Zahlenwert zurück, mit dem er mitteilen kann, ob er gelungen ist. Diesen Zahlenwert nennt man Rückgabewert oder auch Exit-Code. Dabei gilt folgende Konvention:

- 0 bedeutet Erfolg
- 1 und jeder andere Wert bedeutet Misserfolg

Der Benutzer kann diesen Wert aus dem Inhalt der Variablen `$?` entnehmen:

Terminal

```
schueler@debian964:~$ cal August 2023
   August 2023
...
schueler@debian964:~$ echo $?
0
schueler@debian964:~$ cal Heinz 2023
cal: Heinz is neither a month number (1..12) nor a name
schueler@debian964:~$ echo $?
64
```

Bei einem falschen Monatsnamen gibt `cal` also den Wert 64 zurück. Ebenso kann man sehen, ob der `tar`-Befehl gelungen ist (indem man ein Verzeichnis sichern lässt, das es gar nicht gibt):

Terminal

```
schueler@debian964:~$ tar -cvf daten.tar daten # es gibt daten/ nicht
tar: daten: Funktion stat fehlgeschlagen: Datei oder Verzeichnis
     nicht gefunden
tar: Beende mit Fehlerstatus aufgrund vorheriger Fehler
schueler@debian964:~$ echo $?
2
schueler@debian964:~$ mkdir daten/ # deshalb lege ich daten/ jetzt an
schueler@debian964:~$ tar -cvf daten.tar daten
daten/
schueler@debian964:~$ echo $?
0
```

Beim ersten Mal gab es den Rückgabewert 2 (=Fehlschlag); beim zweiten Mal den Rückgabewert 0 (=Erfolg).

Die Verzweigung in der Shell nutzt den Rückgabewert aus:

simplebackup2.sh

```
1 #!/bin/bash
2 if tar -cvf ~/daten.tar ~/daten/
3 then
4    echo "Das war ein Erfolg"
5 else
6    echo "Das war wohl nix"
7 fi
```

- Zeile 2: Hinter dem Schlüsselwort `if` steht hier kein Vergleichsausdruck (wie in vielen anderen Sprachen), sondern eine komplette Befehlszeile. Die wird ausgeführt.
- Zeile 3: Falls der Befehl aus Zeile 2 erfolgreich war und deshalb den Rückgabewert 0 zurückgegeben hat, werden die Befehle zwischen `then` und `else` ausgeführt. Diesen Bereich nennt man den `if`-Zweig.
- Zeile 4: Nur eine Meldung, das soll für den Anfang reichen. Hier kommen später sinnvolle Befehle hin.
- Zeile 5: Falls der Befehl aus Zeile 2 keinen Erfolg hatte und er deshalb einen Wert ungleich 0 zurückgegeben hat, dann werden die Befehle zwischen `else` und `fi` ausgeführt. Diesen Bereich nennt man den `else`-Zweig.
- Zeile 6: Wieder nur eine Meldung ausgeben.
- Zeile 7: Hier endet die Verzweigung. Die beiden Zweige laufen wieder zusammen.

Die Ausführung ergibt:

Terminal

```
schueler@debian964:~$ simplebackup2.sh     # muesste klappen
tar: Entferne führende „/“ von Elementnamen
/home/schueler/daten/
Das war ein Erfolg
schueler@debian964:~$ mv ~/daten ~/garten # nehme das Verzeichnis weg
schueler@debian964:~$ simplebackup2.sh     # muesste schiefgehen
tar: Entferne führende „/“ von Elementnamen
tar: /home/schueler/daten: Funktion stat fehlgeschlagen:
     Datei oder Verzeichnis nicht gefunden
tar: Beende mit Fehlerstatus aufgrund vorheriger Fehler
Das war wohl nix
schueler@debian964:~$ mv ~/garten ~/daten # raeume Verz. wieder hin
```

Die Einrückung in den Zeilen 4 und 6 ist nicht nötig, aber hilfreich (wie in den meisten Programmiersprachen).

Zurück zum oben gezeigten Beispiel. Hier sollte der zweite Befehl `gzip` nur dann ausgeführt werden, wenn der erste Befehl geklappt hat. Das kann dann so aussehen:

simplebackup3.sh

```
1 #!/bin/bash
2 if tar -cvf ~/daten.tar ~/daten/ &> /dev/null
3 then
```

```
4     gzip ~/daten.tar
5  else
6     echo "tar-Befehl fehlgeschlagen"
7     exit 1
8  fi
9  exit 0
```

- Zeile 2: Damit man nicht durch die Ausgabe von `tar` irritiert wird, leitet man sie um in die Mülltonne `/dev/null`.
- Zeile 4: `tar` hat geklappt. Man kann in Ruhe komprimieren.
- Zeile 6: Fehlermeldung für den Benutzer
- Zeile 7: Ende mit Rückgabe eines Fehlercodes. Ja, auch Skripte sollten einen Wert ungleich 0 zurückgeben, wenn sie nicht erfolgreich waren
- Zeile 9: Ende mit Rückgabe des Werts 0 (=Erfolg).

2.4.3 Ergänzungen

Bei der Entwicklung eines Skriptes kann man eine Verzweigung auch interaktiv ausführen lassen. Bei interaktiver Benutzung erhält man, solange die Verzweigung noch nicht fertig ist, eine verkürzte Eingabeaufforderung (meistens das Größer-Zeichen, festgelegt in `$PS2`). Nach der Eingabe von `fi` gelangt man wieder auf die normale Befehlszeile:

Terminal

```
schueler@debian964:~$ if ls *.txt
> then
>    echo "Es gibt hier Textdateien"
> else
>    echo "Keine Textdateien zu sehen"
> fi
Es gibt hier Textdateien
schueler@debian964:~$
```

Mit dem Ausrufezeichen kann man den Rückgabewert einer Befehlsliste invertieren: Aus 0 wird 1, und aus 1, 2, 3, ... wird 0. Damit tauschen der `if`- und der `else`-Zweig ihre Inhalte:

simplebackup4.sh

```
1  #!/bin/bash
2  if ! tar -cvf ~/daten.tar ~/daten/ &> /dev/null
3  then
4     echo "tar-Befehl fehlgeschlagen"
5     exit 1
6  else
7     gzip ~/daten.tar
8  fi
9  exit 0
```

- Zeile 2: Man beachte das Ausrufezeichen vor der Befehlszeile.
- Zeilen 4 und 5: Hier ist die Fehlerbehandlung im `if`-Zweig. Das Programm wird vorzeitig verlassen
- Zeile 7: Hier ist der Kompressions-Befhle im `else`-Zweig

In der Verzweigung darf man den `else`-Zweig übrigens weglassen. Dann kann man dasselbe Skript auch so schreiben:

simplebackup5.sh

```
1 #!/bin/bash
2 if ! tar -cvf ~/daten.tar ~/daten/ &> /dev/null
3 then
4     echo "tar-Befehl fehlgeschlagen"
5     exit 1
6 fi
7 gzip ~/daten.tar
8 exit 0
```

- Zeile 7: Weil bei einem Fehlschlag das Skript in Zeile 5 verlassen wird, werden die Zeilen nach `fi` nur ausgeführt, falls der `tar`-Befehl erfolgreich war. Deshalb kann man den `gzip`-Befehl auch hinter die Verzweigung stellen.

Für Spezialisten gibt es in der Verzweigung auch noch das Schlüsselwort `elif`, mit dem man Ketten aus `if` und `else` zusammenbauen kann.

2.4.4 Das Programm `test`

Mit der Verzweigung allein kann man nur herausfinden, ob ein bestimmter Befehl funktioniert hat. Manchmal fragt man sich aber:

- Existiert die Datei mit dem Namen `xyz.txt`?
- Sind die Strings `"${ABC}"` und `"${DEF}"` gleich?
- Ist die Zahl `${X}` gleich 42?
-

Das Programm `test` gibt uns Antwort auf alle (diese) Fragen. Es antwortet aber nicht in Textform, sondern mit seinem Rückgabewert. Der ist entweder 0, was JA bedeutet, oder er ist 1, was NEIN bedeutet. Hier ein Beispiel:

Terminal

```
schueler@debian964:~$ test 7 = 3 # Leerzeichen beachten!
schueler@debian964:~$ echo $?
1
schueler@debian964:~$ test 7 = 7
schueler@debian964:~$ echo $?
0
```

`test` arbeitet hervorragend mit der Verzweigung zusammen:

Terminal

```
schueler@debian964:~$ if test 7 = 7
> then
>     echo "sind gleich"
> else
>     echo "sind ungleich"
> fi
sind gleich
```

Sinnvoll wird das dann, wenn man Variablen benutzt:

Terminal

```
schueler@debian964:~$ X=3
schueler@debian964:~$ Y=5
schueler@debian964:~$ if test "$X" = "$Y"
> then>    echo "gleich"> else>    echo "ungleich"> fi
ungleich
```

Übrigens ist das `test`-Programm auch unter dem Namen `[` (eckige Klammer auf) benutzbar. Man kann also auch schreiben:

Terminal

```
schueler@debian964:~$ if [ "$X" = "$Y" ]
> then>    echo "gleich"> else>    echo "ungleich"> fi
ungleich
```

Wenn das `test`-Programm mit dem Namen `[` aufgerufen wurde, dann erwartet es am Ende der Befehlszeile als letztes Argument das Zeichen `]` (eckige Klammer zu). Das dient nur dazu, dass Skripte (oder Befehlszeilen) schöner aussehen:

testbeispiel1.sh

```
1 #!/bin/bash
2 X=$1
3 Y=$2
4 if [ "$X" = "$Y" ]
5 then
6    echo "gleich"
7 else
8    echo "ungleich"
9 fi
```

Mit dem `test`-Programm kann man ganz verschiedenartige Tests vornehmen:

a) Tests für Zeichenketten

b) Tests für Zahlenwerte

c) Tests für Dateien und Verzeichnisse

2.4.4.1 `test`-Parameter für Zeichenketten

- `test "$X" = "$Y"` – bedeutet: `"$X"` und `"$Y"` sind gleich
- `test "$X" != "$Y"` – bedeutet: `"$X"` und `"$Y"` sind ungleich
- `test "$X"` – bedeutet: `"$X"` ist keine leere Zeichenkette
- `test -z "$X"` – bedeutet: `"$X"` ist eine leere Zeichenkette

2.4.4.2 `test`-Parameter für Zahlenwerte

- `test "$X" -eq "$Y"` – bedeutet: `"$X"` ist gleich `"$Y"`
- `test "$X" -ne "$Y"` – bedeutet: `"$X"` ist ungleich `"$Y"`
- `test "$X" -lt "$Y"` – bedeutet: `"$X"` ist kleiner als `"$Y"`
- `test "$X" -gt "$Y"` – bedeutet: `"$X"` ist größer als `"$Y"`
- `test "$X" -lt "$Y"` – bedeutet: `"$X"` ist kleiner oder gleich `"$Y"`
- `test "$X" -gt "$Y"` – bedeutet: `"$X"` ist größer oder gleich `"$Y"`

2.4.4.3 `test`-Parameter für Dateisystemobjekte

- `test -e "$X"` –bedeutet: `"$X"` ist der Pfadname eines existierenden Objekts
- `test -f "$X"` –bedeutet: `"$X"` ist der Pfadname einer Datei
- `test -d "$X"` –bedeutet: `"$X"` ist der Pfadname eines Verzeichnisses
- und viele weitere Optionen ...

2.4.5 Skript mit `test`-Programm

Mit dem `test`-Programm kann man das einfache Backup-Skript weiter aufwerten:

simplebackup6.sh

```
1  #!/bin/bash
2  if ! test -d ~/daten/
3  then
4      echo "Zu sicherndes Verzeichnis existiert nicht"
5      exit 2
6  fi
7  if ! tar -cvf ~/daten.tar ~/daten/ &> /dev/null
8  then
9      echo "tar-Befehl fehlgeschlagen"
10     exit 1
11 fi
12 gzip ~/daten.tar
13 exit 0
```

- Zeile 2: Abfrage, ob das Verzeichnis `daten` existiert
- Zeile 5: Es ist sinnvoll, wie hier für jeden Fehlerfall einen anderen Rückgabewert zu verwenden. Dann weiß der Benutzer sofort, *was* fehlgeschlagen ist, und man kann als Administrator schneller helfen.