# Übersicht Shell-Scripten

**!!!! Wichtig: Bei Shell-Scripten enden die Zeilen nicht mit einem Strichpunkt !!!!**

## Erste Zeile eines Shell-Scripts:

#! /bin/bash

## Variablen in Shell-Scripts:

Variablennamen müssen mit einem Buchstaben beginnen, können Ziffern und '_' enthalten, maximal 14 Zeichen lang sein und werden nach einer Konvention immer groß geschrieben.

### Inhalt:

Variablen können Strings oder ganze Zahlen enthalten

### Zuweisung:

VARIABLENNAME=Wert bzw. VARIABLENNAME="String"

### Ausgabe:

echo $VARIABLENNAME

**Sichere Ausgabe:** (funktioniert auch mit leeren Variablen)

echo "$VARIABLENNAME"

**Quoting:** (deaktivieren von nachfolgendem Sonderzeichen wie z.B. $ oder '')

Mit einem Backslash wird das nachfolgende Sonderzeichen deaktiviert.
Der Befehl echo \$VARIABLENNAME erzeugt die Ausgabe „$VARIABLENNAME"
und gibt nicht den Inhalt der Variable VARIABLENNAME aus.

**Backticks:** `.....` (backward pipe)

Mit den Backticks wird eine sogenannte „backward pipe" ausgeführt, d.h.
es wird erst der Befehl innerhalb der Backticks ausgeführt und das Ergebnis dieses Befehls an die Variable übergeben.

z.B. wird bei dem Befehl VARIABLENNAME=`whoami` zuerst der Befehl whoami ausgeführt, der die Kennung des aktuell angemeldeten Users zurückgibt und dann dieses Ergebnis der Variable VARIABLENNAME zugewiesen.

## Mathematik in Shell-Scripten:

Berechnungen im ganzzahligen Bereich werden mit dem Ausdruck „expr" durchgeführt.
Er kann addieren ( + ), subtrahieren ( - ), multiplizieren ( * ) und dividieren ( / )

Beispiel:

```
ERGEBNIS=`expr $SUMMAND1 + $SUMMAND2`
ERGEBNIS=`expr $SUMMAND1 * 5`
```

# Übersicht Shell-Scripten

## Bedingungstester: „test“ (für if- und while-Schleifen)

Bei if- und while-Schleifen ist eine Bedingung gefordert, um bestimmte Anweisungen auszuführen. Diese Bedingungen werden mit dem Befehl „test“ getestet.

### *Eigenschaften von Dateien testen:*

**Name im Verzeichnis vorhanden, ist File:** ( Parameter –f )

Beispiel:

```
if test –f $FILE then ....
```

Hier wird getestet, ob der Name, der in der Variable FILE gespeichert ist, im aktuellen Verzeichnis vorhanden ist und ob er ein File (Datei) ist

**Name im Verzeichnis vorhanden, ist File und nicht leer:** ( Parameter –s )

Beispiel:

```
if test –s $DATEI then ....
```

Hier wird getestet, ob der Name, der in der Variable DATEI gespeichert ist, im aktuellen Verzeichnis vorhanden ist, ob er ein File ist und ob es nicht leer ist.

**Name im Verzeichnis vorhanden, ist Verzeichnis:** ( Parameter –d )

Beispiel:

```
if test –d $DIR then ....
```

Hier wird getestet, ob der Name, der in der Variable DIR gespeichert ist, im aktuellen Verzeichnis vorhanden ist und ob er ein Verzeichnis ist.

### *Eigenschaften von Zeichenketten testen:*

**String1 gleich String 2:** ( Parameter = )

Beispiel:

```
if test $TEXT1 = $TEXT2 then ....        oder        if test $TEXT1 = “String” then ....
```

Hier wird getestet, ob die beiden Strings gleich sind.

**String1 nicht gleich String2:** ( Parameter != )

Beispiel:

```
if test $TEXT1 != $TEXT2 then ....       oder        if test $TEXT1 != “String” then ....
```

Hier wird getestet, ob die beiden Strings nicht gleich sind.

**String leer:** ( Parameter –z )

Beispiel:

```
if test -z $TEXT1 then ....
```

Hier wird getestet, ob die Variable TEXT1 leer ist.

# Übersicht Shell-Scripten

## Fortsetzung Bedingungstester:

***Eigenschaften von ganzen Zahlen testen:***

Parameter –ne : ungleich
Parameter –eq : gleich
Parameter –ge : grösser oder gleich
Parameter –gt : grösser
Parameter –le : kleiner oder gleich
Parameter –lt : kleiner

Beispiel:

if test $ZAHL1 -gt $ZAHL2 then .... oder if test $ZAHL –lt 20 then ....

Hier wird getestet, ob der Wert der Variable ZAHL1 grösser ist als der Wert der Variable ZAHL2 bzw. ob der Wert der Variable ZAHL kleiner als 20 ist.

**Verknüpfung von Bedingungen:**

Die möglichen Testbedingungen können mit folgenden Operatoren verknüpft werden.

-o für eine ODER-Verknüpfung (OR)
-a für eine UND-Verknüpfung (AND)
! für eine NICHT-Verknüpfung (NOT)

Es ist unter Umständen sinnvoll, die einzelnen Testbedingungen in Klammern zu setzen.

Beispiel:

while test ( $SCHLEIFE –ne 0) –a ( $ENDE != „JA“) then ...

Die while-Schleife wird solange ausgeführt, solange der Wert der Variablen SCHLEIFE ungleich 0 ist und der Wert der Variable ENDE ungleich „JA“ ist.

# Übersicht Shell-Scripten

## Bedingte Ausführungen:

### if-Struktur:

```
if test $VARIABLE = Wert
  then
    Anweisungen bei erfüllter Bedingung
    .....
  else
    Anweisungen bei nicht erfüllter Bedingung
    .....
fi
```

Der else-Zweig kann wegfallen.
Die if-Struktur wird immer mit einem „fi“ beendet.

### while-Struktur:

```
while test $VARIABLE = Wert
  do
     Anweisungen
     ....
  done
```

Die Anweisungen werden solange ausgeführt, bis die Bedingung nicht mehr erfüllt ist.
Die Anweisungen der while-Struktur werden immer zwischen einem „do“ und einem „done“ eingeschlossen.

### for-Struktur:

```
for VARIABLE in Liste mit Werten
  do
     Anweisungen
     ....
  done
```

Die Anweisungen werden für jedes Element in der Liste mit Werten ausgeführt. Dabei enthält VARIABLE das aktuelle Listenelement.

Beispiel:

```
for NAME in Stefan Hans Peter Tom Sepp
  do
    echo “Name: “
    echo $NAME
  done
```

Diese Schleife gibt nacheinander die Namen aus.

# Übersicht Shell-Scripten

## Positionsparameter:

Wenn man längere Ausgaben bearbeiten muss, kann man dazu die Positionsparameter nutzen.

Um eine Ausgabe in die Pufferspeicher der Positionsparameter zu schreiben wird der Befehl „set“ benutzt.

Beispiel: set `ls –l`

Es gibt 9 Positionsparameter von $1 bis $9. Zusätzlich wird die Anzahl der ausgegebenen Zeilen in der Variable $# gespeichert.

Die Positionsparameter beginnen immer am Anfang des Puffers. Um nun das Fenster mit den 9 Positionsparametern zu verschieben, wird der Befehl „shift“ benutzt.

Beispiel:

| | |
|---|---|
| set `ls –l` | Einlesen in den Puffer |
| echo $9 | Ausgeben des Positionsparameters $9 (Wert z.B. 20) |
| shift 8 | Verschieben des Positionsrahmenfensters |
| echo $1 | Ausgeben des Positionsparameters $1<br>( Jetzt gleicher Wert wie vorher bei $9 = 20) |

Sobald im Puffer ein Leerzeichen (Blank) auftritt, wird der nächste Positionsparameter gefüllt.

Beispiel:

Pufferinhalt: „Dies ist eine Uhrzeit: 12:29“
Inhalt Pos.-Paramter $1: „Dies“
Inhalt Pos.-Paramter $2: „ist“
Inhalt Pos.-Paramter $3: „eine“
Inhalt Pos.-Paramter $4: „Uhrzeit:“
Inhalt Pos.-Paramter $5: „12:29“

## Dateihandling – Dateierzeugung:

Die Ausgabe eines Befehls kann in eine Datei umgeleitet werden. Dies geschieht mit folgenden Operatoren:

**Datei überschreiben bzw. neu anlegen:** ( Operator > )

Beispiel: echo „Text“ > „Dateiname“

Schreibt den String „Text“ in die Datei „Dateiname“. Wenn die Datei vorhanden ist, wird sie überschrieben.

**An Datei anhängen bzw. neu anlegen:** ( Operator >> )

Beispiel: echo „Text“ >> „Dateiname“

Hängt den String „Text“ in die Datei „Dateiname“ hinten an. Wenn die Datei nicht vorhanden ist, wird sie angelegt.