bash-Skriptprogrammierung

1 Angabe der Shell

`#!/bin/bash` muss ganz links in der ersten Zeile stehen

2 Kommentare

`# Kommentar` alles, was hinter einem `#` steht, wird ignoriert

3 Variablen

3.1 Wertzuweisung

`VARIABLE='Irgendein Text'`	Zuweisung von Text
`VARIABLE=42`	Zuweisung einer ganzen Zahl
`VARIABLE=$( KOMMANDO )`	Ausgabe des Kommandos in Variable speichern
`read VARIABLE`	Benutzereingabe von Konsole lesen

3.2 Inhalt lesen

`$VARIABLE` oder `${VARIABLE}` Inhalt der Variablen lesen

4 Rechnen mit ganzen Zahlen

4.1 Terme mit `$(( ))`

Beispiele `SUMME=$(( ZAHL1 + ZAHL2 ))` oder `SUMME=$(( $ZAHL1 + $ZAHL2 ))`

Rechenoperatoren `+ - * / % **` (Potenz)

4.2 Anweisungen mit `(( ))`

Beispiel `(( x = x + 3 ))`

Rechen- und Zuweisungsoperatoren `+ - * / % ** = += -= *= /= %= ++ --`

5 Skript beenden

`exit 0`	kein Fehler aufgetreten
`exit 1`	Fehler aufgetreten

6 Rückgabewert des letzten Kommandos

`$?` oder `${?}`

7 Parameter

`$#` oder `${#}`	Anzahl der übergebenen Argumente
`$0` oder `${0}`	Dateiname des Skripts
`$1` oder `${1}`	1.Parameter
...	
`$9` oder `${9}`	9.Parameter
`$*` oder `${*}`	alle übergebenen Parameter

8 Bedingungen

0 = true, 1 = false

8.1 Zeichenketten

`[ $TEXT ]`	TEXT nicht leer
`[ -z $TEXT ]`	TEXT leer
`[ $TEXT1 = $TEXT2 ]`	TEXT1 und TEXT2 identisch
`[ $TEXT1 != $TEXT2 ]`	TEXT1 und TEXT2 nicht identisch

8.2 Dateien und Verzeichnisse

`[ -e $PFAD ]`	Datei existiert
`[ -f $PFAD ]`	ist normale Datei
`[ -d $PFAD ]`	ist Verzeichnis
`[ -r $PFAD ]`	ist lesbar
`[ -w $PFAD ]`	ist schreibbar
`[ -x $PFAD ]`	ist ausführbar
`[ $PFAD1 -nt $PFAD2 ]`	Datei1 ist neuer als Datei2

8.3 Zahlen

`[ $ZAHL1 -eq $ZAHL2 ]`	==
`[ $ZAHL1 -ne $ZAHL2 ]`	!=
`[ $ZAHL1 -gt $ZAHL2 ]`	>
`[ $ZAHL1 -ge $ZAHL2 ]`	>=
`[ $ZAHL1 -lt $ZAHL2 ]`	<
`[ $ZAHL1 -le $ZAHL2 ]`	<=

8.4 Verknüpfung von Bedingungen

`[ Bedingung1 ] || [ Bedingung2 ]`	ODER
`[ Bedingung1 ] && [ Bedingung2 ]`	UND
`! [ Bedingung ]`	NICHT

9 Verzweigungen

9.1 if

```
if [ Bedingung1 ]; then
     Kommandos1
elif [ Bedingung2 ]; then
     Kommandos2
else
     Kommandos2
fi
```

9.2 case

```
case $VARIABLE in
     Wert1)
          Kommandos1
          ;;
     Wert2)
          Kommandos2
          ;;
     *)
          Kommandos3
          ;;
esac
```

10 Schleifen

10.1 for

```
for VARIABLE in Wert1 Wert2 Wert3; do
     Kommandos
done
```

10.2 for

```
for(( i=0; i<5; i++)); do
     Kommandos
done
```

10.3 while

```
while [ Bedingung ]; do
     Kommandos
done
```

10.4 until

```
until [ Bedingung ]; do
     Kommandos
done
```

11 Arrays

11.1 Wertzuweisung

```
name[0]=Franz
name[1]=Kohnle
```

11.2 Inhalt lesen

```
${name[1]}
```

12 Funktionen

12.1 Definition (vor dem Aufruf)

```
funktionsname(){
    Kommando1
    Kommando2 $1 $2 # Parameter
}
```

12.2 Aufruf

```
funktionsname argument1 argument2
```