Betriebssysteme Linux

HA2 Shell-Script 1 Feedback

FH Burgenland

UNIVERSITY OF APPLIED SCIENCES

BRINGT BESONDERES ZUSAMMEN

DI Franz Knipp
WS 2020/21

Iterieren über Argumente

Für das Iterieren über alle Argumente ist die **kürzeste Variante:**

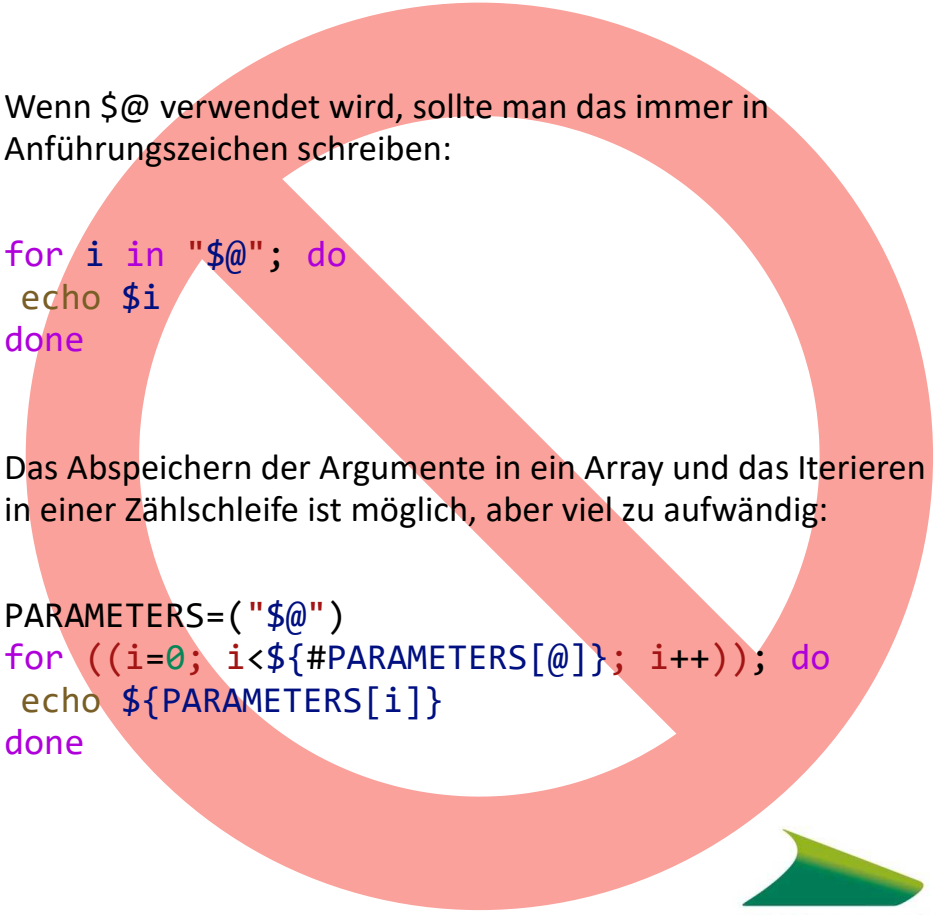
```
for i ; do
 echo "$i"
done
```

Wenn $@ verwendet wird, sollte man das immer in Anführungszeichen schreiben:

```
for i in "$@"; do
 echo $i
done
```

Das Abspeichern der Argumente in ein Array und das Iterieren in einer Zählschleife ist möglich, aber viel zu aufwändig:

```
PARAMETERS=("$@")
for ((i=0; i<${#PARAMETERS[@]}; i++)); do
 echo ${PARAMETERS[i]}
done
```



FH Burgenland
UNIVERSITY OF APPLIED SCIENCES

Argumente in Anführungszeichen

Setzen Sie prinzipiell **alle Argumente in Anführungszeichen**, außer das Zerlegen in einzelne Wörter bzw. das Expandieren ist explizit erwünscht.

Aufruf des Skripts mit `./test-script "123 *"`

Im Skript: `validate $1`

Expandiert zu: `validate 123 *`

Und weiter zu: `validate 123 erste-datei zweite-datei`

Die Funktion `validate` erhält dann drei Parameter.

Korrekt: `validate "$1"`

Expandiert zu: `validate "123 *"`

FH Burgenland
UNIVERSITY OF APPLIED SCIENCES

Argumente in […]

Die Bash führt bei der Verwendung von […] ein Word Splitting und Expandieren der Dateinamen (z. B. beim *) durch.

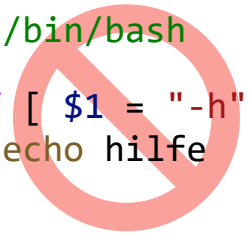
Lösung 1: **Anführungszeichen**

```
if [ "$1" = "-h" ]; then
  echo hilfe
fi
```

Lösung 2: **Verwendung von [[…]]**

```
if [[ $1 = "-h" ]]; then
  echo hilfe
fi
```

Probieren Sie folgendes Skript und die daraus resultierenden Fehler:



```
#!/bin/bash

if [ $1 = "-h" ]; then
  echo hilfe
fi
```

```
> ./test-if
./test-if: line 3: [: =: unary operator expected
> ./test-if " "
./test-if: line 3: [: =: unary operator expected
> ./test-if *
./test-if: line 3: [: too many arguments
```

FH Burgenland
UNIVERSITY OF APPLIED SCIENCES