

LU07b - Parameter und Rückgabewerte

ShellProg - Kapitel 6.3-6.4

Parameter in Funktionen Bash-Funktionen können Argumente annehmen, ähnlich wie Skripte. Diese Argumente werden innerhalb der Funktion über Positionsparameter referenziert:

- ``$1`` für das erste Argument, ``$2`` für das zweite usw. - ``$@`` und ``$*`` enthalten alle Argumente. - ``$#`` gibt die Anzahl der übergebenen Argumente zurück.

Beispiel:

```
#!/bin/bash

addiere() {
    echo "$(($1 + $2))"
}

addiere 5 10 # Aufruf der Funktion mit zwei Argumenten
```

Ausgabe: `</code> 15 </code>`

In diesem Beispiel sind ``$1`` und ``$2`` die ersten beiden Argumente, die an die Funktion ``addiere`` übergeben wurden.

Parameter prüfen Du kannst prüfen, ob Argumente übergeben wurden:

```
zeige_argumente() {
    if [ "$#" -eq 0 ]; then
        echo "Keine Argumente übergeben."
    else
        echo "Argumente: $@"
    fi
}

zeige_argumente Hallo Welt
```

Ausgabe: `</code> Argumente: Hallo Welt </code>`

Rückgabewerte Bash-Funktionen geben standardmäßig numerische Rückgabewerte zwischen ``0`` und ``255`` zurück, die mit ``return`` gesetzt werden können. Diese Werte können über die Variable ``$?`` nach dem Funktionsaufruf abgerufen werden.

Numerischer Rückgabewert:

```
#!/bin/bash
```

```
prüfe_zahl() {  
    if [ $1 -gt 10 ]; then  
        return 0 # Erfolg  
    else  
        return 1 # Fehler  
    fi  
}  
  
prüfe_zahl 15  
if [ $? -eq 0 ]; then  
    echo "Die Zahl ist größer als 10."  
else  
    echo "Die Zahl ist 10 oder kleiner."  
fi
```

Ausgabe: `</code> Die Zahl ist größer als 10. </code>`

Hinweis: ``return`` ist auf numerische Werte beschränkt. Komplexe Daten müssen über andere Wege „zurückgegeben“ werden.

—

Nicht-numerische Rückgabewerte Nicht-numerische Daten (wie Strings oder Arrays) können indirekt zurückgegeben werden, z. B. durch ``echo`` und Command Substitution (``$(...)``).

Beispiel mit ``echo``:

```
#!/bin/bash  
  
get_date() {  
    echo  $$(date)$   
}  
  
aktuelles_datum=$(get_date)  
echo "Aktuelles Datum:  $$(aktuelles_datum)$ "
```

Ausgabe: `</code> Aktuelles Datum: Mo 26. Nov 2024 10:15:30 CET </code>`

—

Rückgabe über globale Variablen Eine weitere Möglichkeit besteht darin, globale Variablen zu nutzen, um Daten aus einer Funktion verfügbar zu machen.

Beispiel:

```
#!/bin/bash  
  
berechne() {  
    ergebnis=$(( $1 * $2 ))  
}
```

```
berechne 4 5  
echo "Das Ergebnis ist: $ergebnis"
```

Ausgabe: `</code> Das Ergebnis ist: 20 </code>`

—

Zusammenfassung - **Parameter:** Werden mit `\$1`, `\$2`, usw. referenziert und mit `@\$` oder `*\$` gesammelt. - **Rückgabewerte:** Numerische Werte über `return`, komplexe Daten über `echo` oder globale Variablen. - Diese Mechanismen machen Bash-Funktionen flexibel und ermöglichen die Verarbeitung verschiedener Arten von Daten.

From:

<https://wiki.bzz.ch/> - **BZZ - Modulwiki**

Permanent link:

<https://wiki.bzz.ch/modul/m122/learningunits/lu07/schnittstelle?rev=1732599714>

Last update: **2024/11/26 06:41**

