

Shell-Programmierung

Parameterübergabe an Shell-Skripte

Um auf die an ein Shellskript übergebenen Parameter zugreifen zu können, wurden intern von der Shell spezielle Variablen vordefiniert:

| | |
|-----------|--|
| \$* | Gibt Zugriff auf alle dem Skript übergebenen Parameter. Der Rückgabewert ist eine Liste, die sich z.B. für die Übergabe an eine <code>for</code> -Schleife eignet. |
| \$1 - \$n | Gibt Zugriff auf den n -ten übergebenen Parameter |
| \$0 | Enthält den Programmnamen, wie er vom Benutzer zum starten des Skripts angegeben wurde |

Falls man in einem Skript einen Parameter nach dem anderen abarbeiten will, bietet sich an, nur auf \$1 zuzugreifen und nach der Bearbeitung die Parameterliste mit dem Befehl `shift` um eine Position zu verschieben.

Arrays

Ein Array wird von der Shell erstellt, sobald man einer Variable mit dem Syntax `variable[index]=wert` einen Wert zuweist. Hierbei muss $index \geq 0$ sein. Beim Zugriff auf eine so definierte Variable ist zu beachten, dass der Variablenname mit dem `index` in geschweifte Klammern (`{}`) eingeschlossen ist.

Beispiel:

```
zahl[1]=5
zahl[2]=9
zahl[3]=276
echo ${zahl[2]}    Ausgabe: 9
```

Der Aufruf `${variable[*]}` gibt den Inhalt aller Elemente des Arrays als Liste zurück.

Prozeduren, Funktionen

```
Syntax: function name() {...}
oder:   name() {...}
```

Auf die an eine Funktion übergebenen Parameter greift man genauso zu, als ob diese direkt an das Skript übergeben wurden.

Variablen gelten normalerweise immer global, so dass man ihren Wert sowohl aus dem Hauptprogramm als auch aus einer Funktion heraus ändern kann. Will man dies verhindern (um z.B. Rekursionen zu ermöglichen), muss man eine Variable in der Funktion lokal definieren. Dies geschieht mit: `local variable1 variable2 ...`

Rekursion

Unter Rekursion versteht man, dass sich eine Funktion selber wieder aufruft, um der Lösung eines Problems näher zu kommen. Ein gutes Beispiel ist die Fakultätsfunktion, die das Produkt aus allen Zahlen \mathbb{N}_0 bis zu einer vorgegebenen Zahl berechnet, wobei die Fakultät von 0 auf 1 definiert wurde:

$$4! = 4 * 3 * 2 * 1$$

Man kann dies aber auch folgendermassen sehen:

$$4! = 4 * 3! = 4 * 3 * 2! = 4 * 3 * 2 * 1! = 4 * 3 * 2 * 1 * 0! = 4 * 3 * 2 * 1 * 1 = 24$$

Fakultätsfunktion:

```
fact()
{
    local zahl
    zahl=$1

    if [ "$1" = "0" ]; then
        echo 1
    else
        echo ${zahl}*`fact ${zahl-1}`
    fi
}
```

Beispielprogramm (Türme von Hanoi):

```
#!/bin/sh
#Tuerme von Hanoi als Shell-Skript

#Bewegen der Tuerme von Hanoi
#Syntax:
#bewegehanoi scheiben von nach extra
function bewegehanoi()
{
    #Lokale Definition der Variablen:
    local scheiben von nach extra

    #Zuweisen der uebergebenen Variablen:
    scheiben=$1
    von=$2
    nach=$3
    extra=$4

    if [ "$scheiben" = "1" ]; then
        echo "Bewege oberste Scheibe von $von nach $nach"
    else
        #Alle Scheiben bis auf eine auf den Extra-Stapel schieben
        bewegehanoi ${scheiben-1} $von $extra $nach
        #Eine Scheibe auf den Ziel-Stapel schieben
        bewegehanoi 1 $von $nach $extra
        #Restliche Scheiben vom Extra- auf den Zielstapel schieben
        bewegehanoi ${scheiben-1} $extra $nach $von
    fi
}

if [ "$1" != "" ]; then
    bewegehanoi $1 1 3 2
else
    echo "Syntax: $0 anzahl-scheiben"
fi
```