2.7 Skripte/Funktionen

2.7.1 Shellskript-Funktionen

In einem Shellskript kann man auch eine Funktion definieren und aufrufen. Dies geschieht in der aktuellen Shell! Es ist also nicht immer nötig (aber möglich), für diesen Zweck eine eigene Shellskript-Datei zu erstellen und (mit einer extra-Shell) aufzurufen.

Die Form der Definition ist folgendermaßen:

```
function FN() { BL; }
```

Seit einiger Zeit wird in der Grammatik der Bash statt der Befehlsliste `BL` ein Verbundbefehl angegeben. Damit darf eine Funktion anstelle von Befehlen z.B. auch eine in doppelte runde Klammern eingeschlossene Berechnung enthalten.

Das Schlüsselwort `function` kann weggelassen werden. Der Aufruf erfolgt durch den Funktionsnamen:

Terminal

```
schueler@debian964:~$ function willi() { echo "ich bin willi"; }
schueler@debian964:~$ willi
ich bin willi
schueler@debian964:~$
```

Die geschweiften Klammern sind reservierte Worte (wie `for` oder `if`) und müssen daher von den Befehlen der Befehlsliste durch ein Leerzeichen (oder Zeilenumbruch) getrennt werden. Mit `set` kann man sich die aktuell verfügbaren Funktionen übrigens ansehen.

Will man eine Funktion an aufgerufene Shells (und Shellskripte) exportieren, benötigt man die Option `-f`:

Terminal

```
schueler@debian964:~$ export -f willi
```

2.7.2 Shellskript-Funktionen und Variable

Alle Shell- und Umgebungsvariablen sind in einer Shellskript-Funktion verfügbar. Sie sind hier also wie globale Variablen in Programmiersprachen anzusehen (von der Verwendung ist eher abzuraten).

Die Übergabe von Parametern an eine Funktion erfolgt genauso wie an ein Shellskript. In der Funktion sind die Parameter `$1`, `$2`, `$3`, ... des Shellskripts ersetzt durch die Aufrufparameter der Funktion.

Innerhalb einer Funktion kann mit dem `local`-Befehl eine lokale Variable angelegt werden:

```
1  function zaehle() # $1=von, $2=bis
2  {
3      local -i i # lokale int-Variable i
4
5      for((i="$1";i<"$2";++i))
6      do
7          ping -c1 "192.168.1.$i"
8      done
9      return 0
10 }
```

Der `return`-Befehl bewirkt in der Shellskript-Sprache das Gleiche wie in C und Java: Die Funktion wird an dieser Stelle beendet; optional kann ein (leider nur ganzzahliger) Wert an den Aufrufer zurückgegeben werden, der mit `$?` abgefragt werden kann.

2.7.3 Rekursion bei Shellskript-Funktionen

Bei Shellskript-Funktionen ist auch Rekursion möglich:

```
1  function fakultaet()
2  {
3      if [ "$1" -gt 1 ]
4      then
5          fakultaet $(($1 - 1))
6          return $(($? * $1))
7      else
8          return 1
9      fi
10 }
```

Dann kann man eingeben:

Terminal

```
schueler@debian964:~$ source fakultaet.src
schueler@debian964:~$ fakultaet 4
schueler@debian964:~$ echo $?
24
```