2.3 Skripte/Parameter und Rückgabewerte

2.3.1 Parameter gesucht

Wir können die Programme `mac2.sh` und `mac3.sh` benutzen, aber sie sind umständlich: Entweder muss man vorher die Umgebungsvariable `$IPADR` richtig setzen oder man ist dazu gezwungen, beim Programmlauf den Wert für `$IPADR` einzugeben.

```
1 #!/bin/bash
2 read -p "Bitte IP eingeben: " IPADR
3 ping -c1 -w1 "$IPADR" &> /dev/null
4 /sbin/arp -n | grep "$IPADR" | egrep -o '[0-9a-f:]{17}'
```

[title=mac3.sh]

Wesentlich schöner wäre es, wenn man den Wert schon beim Aufruf auf der Befehlszeile per Parameter übergeben könnte. Dann sähe unser Programm auch gleich professioneller aus.

Was passiert eigentlich, wenn man das einfach tut?

Terminal

```
schueler@debian964:~$ mac3.sh 10.92.0.1
Bitte IP eingeben: ^C
```

Man kann also Parameter eingeben, sie werden nur nicht gelesen, nicht berücksichtigt. Aber wie liest man sie im Skript?

2.3.2 Daten an ein Skript übergeben II: Parameter

Erfahrungsgemäß führt die Benutzung globaler Variablen in der Programmierung zu sehr unübersichtlichen Systemen.

Es ist wesentlich besser, nur die benötigten Daten (und keine anderen) per Parameter an andere Teilsysteme abzugeben. Man nennt dies das *Prinzip der schmalen Datenkopplung*.

Dieses Prinzip findet sich auch in Shell-Skripten wieder: So ist es möglich, einem Skript beliebig viele Kommandozeilen-Parameter mitzugeben. Sie sind zugreifbar über die Bezeichner `$0`, `$1`, `$2` usw.; dabei ist `$0` der Name des Skriptes, `$1` der Name des ersten Parameters, `$2` der Name des zweiten usw. Die Anzahl der Parameter bekommt man in der Variablen `$#`. Hier ein Beispiel:

param.sh

```
1 #!/bin/bash
2 echo '$0=' "$0"
3 echo '$1=' "$1"
4 echo '$2=' "$2"
5 echo '$3=' "$3"
6 echo '$4=' "$4"
7 echo '$@=' "$@"
8 echo '$#=' "$#"
```

Terminal

```
schueler@debian964:~$ param.sh "Bad Salzuflen" "Horn-Bad Meinberg"
$0= ./param.sh
$1= Bad Salzuflen
$2= Horn-Bad Meinberg
$3=
$4=
$@= Bad Salzuflen Horn-Bad Meinberg
$#= 2
```

Zum Testen kann man die Parameter auch von Hand setzen. Dazu dient der `set`-Befehl:

Terminal

```
schueler@debian964:~$ set Hund Katze Maus
schueler@debian964:~$ echo "$1"
Hund
schueler@debian964:~$ echo "$2"
Katze
schueler@debian964:~$ echo "$3"
Maus
```

Mit `$@` und mit `$*` bekommt man alle Parameter außer dem Skriptnamen `$0`. Dabei unterscheiden sich die beiden Varianten dann, wenn sie maskiert werden. Um das zu verstehen, schaltet man am besten die Shell-Option `-x` ein, bei der jede Befehlszeile vor der Ausführung ausgegeben wird. Der Befehl `:` ist der Null-Befehl, der nichts tut. Interessant ist, wie unterschiedlich die beiden Parameter expandiert werden:

Terminal

```
schueler@debian964:~$ set "Audi Q3" "BMW X1" #Param. $1 und $2 setzen
schueler@debian964:~$ set -x                 #Zeilen ab jetzt anzeigen
schueler@debian964:~$ : $@
+ : Audi Q3 BMW X1          #4 Worte
schueler@debian964:~$ : $*
+ : Audi Q3 BMW X1          #4 Worte
schueler@debian964:~$ : "$@"
+ : 'Audi Q3' 'BMW X1'      #2 Worte
schueler@debian964:~$ : "$*"
+ : 'Audi Q3 BMW X1'        #1 Wort
schueler@debian964:~$ set +x              #Zeilen nicht mehr anzeigen
+ set +x
```

In den meisten Fällen braucht man `"$@"`, um jeden Parameter einzeln zu bearbeiten[1].

2.3.3 Rückgabewert von Shell-Skripten

Das Skript gibt wie andere Programme auch einen Rückgabewert an die aufrufende Shell zurück. Und zwar wird der Rückgabewert des letzten Befehls zum Rückgabewert des Skriptes.

Möchte man einen speziellen Wert zurückgeben, dann benutzt man den `exit`-Befehl:

- Das Skript bricht an dieser Stelle ab
- Es wird der Zahlenwert zurückgegeben, der hinter `exit` genannt ist

Es wurde vereinbart, dass `exit 0` einen fehlerfreien Verlauf angibt. Mit den Werten 1 bis 255 können verschiedene Fehlerarten unterschieden werden:

exitbeispiel.sh

```
1 #!/bin/bash
2 declare -i ZAHL
3 echo -n "Bitte eine Zahl eingeben (0-255): "
4 read ZAHL
5 echo "Jetzt beende ich das Skript mit dem Exit-Code $ZAHL."
6 exit "$ZAHL"
```

[1] Übrigens ist es möglich, auch in Shellskripten Kommandozeilen mit *Optionen* auszuwerten. Allgemein sind Optionen ja Parameter, die mit einem Minuszeichen beginnen. Das Programm `getopt` kann dabei helfen, Optionen von anderen Paramatern zu trennen und zu verarbeiten (dazu später mehr).

Die aufrufende Shell kann den Rückgabewert über den Parameter `$?` abfragen:

Terminal
```
schueler@debian964:~$ exitbeispiel.sh
Bitte eine Zahl eingeben (0-255): 123
Jetzt beende ich das Skript mit dem Exit-Code 123.
schueler@debian964:~$ echo $?
123
```

2.3.4 Parameter gefunden

Jetzt braucht man nur noch das Programm mit der MAC-Adresse in der zweiten Zeile zu modifizieren und kann damit den Parameter einlesen:

```
1 #!/bin/bash
2 IPADR="$1"
3 ping -c1 -w1 "$IPADR" &> /dev/null
4 /sbin/arp -n | grep "$IPADR" | egrep -o '[0-9a-f:]{17}'
```

Und jetzt noch einmal der Aufruf mit einem Parameter:

Terminal
```
schueler@debian964:~$ mac4.sh 10.92.0.1
11.22.33.44.55.66
```

Die Antwort passend zur Anfrage zeigt, dass der Parameter gelesen und verarbeitet wurde.