Schleifen

Eine weitere sehr wichtige und häufig erforderliche Grundstruktur bei der Shell-Programmierung sind Schleifen. wie auch in höheren Programmiersprachen, stehen hierzu zwei verschiedene Schleifentypen zur Verfügung:

1. **for**-Schleifen
2. **while**-Schleifen

Sehen wir uns zunächst die for-Schleife etwas genauer an.

for und zurück – ein erstes Beispiel

Mit einer **for**-Schleife lassen sich hervorragend Befehle mit wechselnden Befehlsparametern ausführen. Wie ist das gemeint? Stellen sie sich vor, dass der auszuführende Befehl der **ping**-Befehl ist, der ja immer einen Parameter benötigt: nämlich den anzupingenden Rechner.

```
#! /bin/bash
clear
echo "Pingtest beginnt...."
echo
for PC in GERAU3 GERAU4 ASTERIX OBELIX; do
    echo "Ein Pingpaket an $PC"
    ping -c1 $PC
done
echo
echo "...und ist abgeschlossen!"
```

Eine frei definierte Variable namens **PC**, die nacheinander die Listenelemente aufnimmt.

Vier Listenelemente:
GERAU3
GERAU4
ASTERIX
OBELIX

Der interessante Teil erscheint nach den beiden ersten **echo**-Zeilen. Dort beginnt die **for**-Schleife. Hinter dem Schlüsselwort **for** folgt eine Variable, deren Name frei wählbar ist, wie z.B. **PC**.

Danach folgt das Schlüsselwort **in**. Anschließend wird die Liste[1] mit allen Elementen angegeben. Die Liste wird durch ein Semikolon abgeschlossen. Diese Kopfzeile der **for**-Schleife endet dann mit dem Schlüsselwort **do**.

Anschließend folgen die Befehle, die nacheinander mit einem der Listenelemente als Parameter ausgeführt werden sollen. Hierbei ist es nun wieder wichtig, das **$**-Zeichen dem Shell-Variablennamen voranzustellen!

Innerhalb der Schleife wird bei der Befehlsausführung für die Variable das jeweilige Listenelement eingesetzt.

Die Schleifenstruktur endet dann mit dem Schlüsselwort **done**.

Übungen 1

1. Erstellen Sie das obige Skript in ihrem Home-Verzeichnis unter dem Namen **ping-test** und setzen Sie das Execute-Recht. Testen Sie das Skript aus (**ASTERIX** sollte nicht erreichbar sein).
2. Erstellen Sie zu Testzwecken drei Text-Dateien mit den Namen **text1**, **text2** und t**ext3**. Schreiben Sie einen kurzen beliebigen Text in jede dieser drei Dateien.
   Programmieren Sie nun ein Shell-Skript, das in jeder dieser drei Dateien mit dem **grep**-Befehl nach einem bestimmten Wort sucht (**grep** "Wort" Datei).
   Lassen Sie auch den Namen der jeweils gerade durchsuchten Datei ausgeben (z.B. "Durchsuche test2...")
3. Modifizieren Sie das erste Shell-Skript so, dass bei einem erfolgreichen Ping eine Meldung in der Art "Zielhost GERAU3 erreicht" oder bei nicht erfolgreichem Ping eine entsprechende Meldung per Umleitung in eine Datei namens **pingtest.txt** geschrieben wird. Diese Datei soll anschließend mit dem **less**-Befehl aus dem Skript heraus angezeigt werden.

   (**Hinweis**: Geben Sie in der Kommandozeile nachfolgende Befehle ein, um eine Test-Bedingung zu finden!)
   ping -c1 OBELIX; echo $?
   ping -c1 ASTERIX; echo $?

1 Die erlaubten Listentrennzeichen sind in der Shell-Variablen **IFS** gespeichert. Mögliche Listentrennzeichen sind i.d.R. das Leerzeichen, der Tabulator (**\t**) und das Newline-Zeichen (**\n**)

OBELIX sollte erreichbar sein; **ASTERIX** sollte nicht erreichbar sein.

Immer weiter `for` – ein zweites Beispiel

Im vorigen Beispiel waren die zu verarbeitenden Listenelemente bereits im Shell-Skript fest definiert. In manchen Fällen mag das bereits ausreichend sein; in anderen Situationen hilft das nicht. Stellen Sie sich vor, dass im zweiten Beispiel nicht nur vier Rechner angepingt werden sollen, sondern der Erreichbarkeitstest soll für -sagen wir mal- hundert Rechner durchgeführt werden. Stellen sie sich weiterhin vor, dass für alle diese Rechner bereits eine Liste mit deren Namen (z.B. die hosts-Datei zur Namenauflösung) existiert.

Dann wäre es doch sicherlich schön, wenn man einfach jeden Rechnernamen aus dieser bereits existierenden Liste nehmen könnte, um die Erreichbarkeit zu testen! Nichts einfacher als das :)

Sehen sie sich hierzu mal das nächste Shell-Skript an:

```
#! /bin/bash
clear
echo "Pingtest beginnt...."
echo
for PC in $(cat pc-namen.txt); do
    echo "Ein Pingpaket an $PC"
    ping -c1 $PC
done
echo
echo "...und ist abgeschlossen!"
```

Das ist der interessante Teil!

Inhalt von **pc-namen.txt**

```
GERAU3
GERAU4
ASTERIX
OBELIX
```

Was ist anders? Nun, vorhin wurde hinter dem Schlüsselwort **in** eine Liste angegeben. Die Elemente dieser Liste wurden dann für jeden Schleifendurchlauf in die Variable PC eingesetzt.

Jetzt wird die **Liste** durch einen Befehl **dynamisch erstellt**. Der Befehl **`cat pc-namen.txt`** gibt normalerweise den Inhalt der angegebenen Datei auf dem Bildschirm aus. Nehmen sie an, in der Datei **`pc-namen.txt`** befinden sich nun die Rechnernamen -jede Zeile ein Rechnername (wie im Bild nebenan).

Der Befehl **`cat`** schreibt nun **nicht** den Inhalt der Datei auf den Bildschirm, sondern liefert die einzelnen Zeilen beim **`for`**-Befehl ab, der nun den Inhalt jeder Zeile für jeden Schleifendurchlauf in die Variable **`PC`** einsetzt.

Das ganze Konstrukt oben nennt man **Kommandosubstitution** (Befehlsersetzung). Erforderlich ist, dass der auszuführende Befehl in den runden Klammern mit einem vorangestellten **$**-Zeichen steht.

Für die Kommandosubstitution gibt es allerdings noch eine zweite Schreibweise, die equivalent ist:

`$(cat pc-namen.txt)` ist in der Funktion identisch mit **`` `cat pc-namen.txt` ``**

Hierbei wird der Befehl von dem Zeichen „**`**“ umschlossen (Neben der **BACKSPACE**-Taste mit **SHIFT**; kommt erst, wenn man noch ein zweites Zeichen eingibt).

Übungen 2

1. Erstellen Sie die Liste mit den obigen Computernamen als Datei **`pc-namen.txt`** und anschließend das obige Skript unter dem Namen **`ping-test2`**. Testen Sie das Skript aus.
2. Ändern sie das Skript ab, in dem sie die zweite Variante der Kommandosubstitution verwenden und testen sie erneut.
3. Diese Übung können Sie erst absolvieren, wenn Sie die nächste Seite bearbeitet haben!

   Erstellen Sie ein Skript, das eine Textdatei **zeilenweise** nach dem Vorkommen einer bestimmten Zeichenfolge (z.B. eine IP-Adresse) untersucht und die Anzahl der Fundstellen zählt. Wenn das Ende der Datei erreicht ist, soll die Anzahl der Fundstellen ausgegeben werden. An das Skript sind zwei Parameter zu übergeben: 1. Dateiname der zu durchsuchenden Datei und 2. die zu suchende Zeichenkette.
   Erstellen Sie hierzu eine geeignete Textdatei zum Testen.

WHILE-Schleife

Die zweite bereits angesprochene Schleifenstruktur ist die **while – do – done** Schleife. Hiermit lassen sich ebenfalls ein oder mehrere Befehle mehrfach ausführen.

```
while Bedingung; do
    Befehle...
done
```

Der Unterschied zur **for**-Schleife liegt darin, dass nun eine **Schleifenbedingung** die Anzahl der Durchläufe steuert. Solange die angegebene Bedingung **WAHR** (**TRUE**) ist, wird die Schleife durchlaufen.

Sobald die Bedingung **FALSCH** (**FALSE**) ist, endet die Schleife und das Shell-Skript wird hinter der **done**-Anweisung fortgesetzt.

Sehen wir uns ein Beispiel an:

```
#! /bin/bash
#Zählschleife für einen Countdown von 10 bis 0

#numerische Shell-Variable COUNT erzeugen
#und mit dem Wert 10 initialisieren
COUNT=10

echo "Countdown läuft...."

while [ COUNT -ge 0 ]; do
    #Aktuellen Wert von COUNT anzeigen
    echo $COUNT
    let COUNT=COUNT-1
    #Alternativ mit Leerzeichen; dann muss der
    #Rechenausdruck in Anführungszeichen stehen
    #let COUNT="COUNT - 1"
done
echo "...wwwwwwuuuuummmmmm"
```

Solange der Wert der Shell-Variablen **COUNT** größer gleich 0 ist, führe die Befehle aus

Durch das Schlüsselwort **let** wird ein arithmetischer Ausdruck eingeleitet

Übung

1. Erstellen sie das obige Shell-Skript unter dem Namen **countdown** (Execute-Recht nicht vergessen!) und testen Sie es aus.
2. Fügen Sie dem Shell-Skript **menue** aus den Übungen zur case-Struktur einen neuen weiteren Menüpunkt zur Programmbeendigung hinzu. Modifizieren Sie das Skript so, dass das Menü nach Auswahl eines Menüpunktes solange erneut angezeigt wird, bis der Menüpunkt zur Beendigung ausgewählt wird.
3. Erstellen Sie das Skript **waitforuser** aus dem Buch "UNIX Shellprogrammierung" auf Seite 99 für die **bash**. Arbeiten Sie zum Testen mit ihrem Nachbarn zusammen, in dem dieser sich ab- und anmeldet.
   Machen Sie sich mit der Arbeitsweise des Skriptes vertraut!
4. Erstellen Sie ein Skript namens **rechnen**, mit dessen Hilfe Sie eine beliebige Folge von Zahlen eingeben können (eine nach der anderen; jeweils mit Enter bestätigt) und das bei Eingabe des Zeichens "=" die Summe der eingegebenen Zahlen auf dem Bildschirm ausgibt.