Hinweise zum Shell Scripting

Kontrollstrukturen in der bash
Die bash unterstützt folgende Kontrollstrukturen:

if / else / fi:
Eine if-Schleife hat eine bedingte Ausführung zur Folge. Syntax:

```
if condition
  then
    statements
[ elif condition2
  then
    statements ]
[ else
  statements ]
fi
```

auch möglich ist:

```
if condition; then
  statements
fi
```

case:
Wählt eine Befehlsfolge in Abhängigkeit von einer Variablen aus. Syntax:

```
case expression in
  pattern1 )
    statements;;
  pattern2 )
    statements;;
  * )
    defaultStatements
esac
```

select:
select baut eine Menüstruktur auf und führt eine Auswahl aus diesem Menü aus.
Syntax:

```
select name [in list]
do
  statements, can use $name
done
```

Tim Wächter
tim.waechter@fh-bielefeld.de

Beispiel:

```
#!/bin/bash
# lese das aktuelle Verzeichnis in
# die Variable DIR
DIR=$(ls)
# Baue ein Menue mit allen Dateien aus:
select CHOICE in $DIR
do
  echo $CHOICE
  if [ -f $CHOICE ]; then
    less $CHOICE
    break
  else
    echo "Select a regular file"
  fi
done
```

for-Schleife:
In einer for-Schleife wird eine Befehlsfolge so oft ausgeführt, wie Elemente in einem Stringarray gefunden werden. Als Variable wird ein Element aus dem Stringarray genommen.
Syntax:

```
for name [in list]
do
  statements
done
```

`list` ist in dem obigen Beispiel ein Array von Strings, `name` ist ein Element dieses Arrays mit dem in der Schleife gearbeitet werden kann. Die Angabe der Liste ist optional. Wird nichts angegeben, so wird als Defaultwert für list das Array `$@` genommen (die Liste der Eingabeparameter eines Skriptes.)

while-Schleife:
In einer while-Schleife wird eine Befehlsfolge wiederholt, solange eine Bedingung erfüllt ist.

Syntax:

```
while condition; do
  statements
done
```

until-Schleife:
In einer until-Schleife wird eine Befehlsfolge wiederholt, bis eine Bedingung erfüllt ist.
Syntax:

```
until condition; do
statements
done
```

while- und until-Schleifen sind also gleichwertig, wenn die condition negiert wird.

conditions:
Die Bedingungen in der if Abfrage und in den while- und until-Schleifen sind Rückgabewerte andere Aufrufe. (Der Rückgabewert eines Aufrufes wird in der Variable `$?` gespeichert.) Deshalb sollten auch Ihre Skripts immer definierte Rückgabewerte besitzen. Im fehlerfreien Fall muss eine 0 zurückgegeben werden, sonst ein Wert ungleich 0.
Wichtig: Das Kommando test bietet eine Vielzahl an eingebauten Tests zu:

- Vergleichen von Strings,
- Auswertung und Behandlung von Dateiattributen,
- arithmetischen Vergleichen.

Siehe dazu Bash Beginners Guide, Kap. 7.1 oder auch das Hilfekommando

```
help test
```

Beispiel: Prüfen, ob die Datei `file1` existiert:

```
test -a file1
echo $?
```

Ist die test-Bedingung erfüllt, existiert also die Datei, wird 0 zurückgegeben, sonst eine 1. (Ausgabe des echo-Befehls.)

Also:

```
if test -a file1; then
  echo file1 existiert.
else
  echo file1 existiert nicht
fi
```

Tim Wächter
tim.waechter@fh-bielefeld.de

Der Befehl test .... kann durch [...] abgekürzt werden.
Also:

```
if test -a file1
```

ist gleichbedeutend mit:

```
if [ -a file1 ]
```

(Achtung: Hinter [und vor] muss ein Leerzeichen stehen, sonst gibt es einen Syntax-Fehler!)
Die Rückgabewerte können logisch verknüpft werden. z.B.:

```
statement1 && statement2 (logische Und-Verknüpfung)
statement1 || statement2 (logische Oder-Verknüpfung)
! statement (Negierung)
```

functions:
Skripte mit vielen Verzweigungen können (ähnlich wie bei Programmen in anderen Programmiersprachen auch) sehr schnell unübersichtlich werden. In einem Bash-Skript können zur Strukturierung von wiederkehrenden Aktivitäten Funktionen gem. dem Motto DRY (Don't Repear Yourself) definiert werden:

```
function myfunc {
  ls -als $1
}
```

und dann verwendet werden:

```
myfunc /bin/bash
```

Funktionen werden in der Shell ausgeführt, in der auch das Skript läuft, das die Funktion beinhaltet. Die formalen Parameter werden in der Funktion mit `$1`, `$2`, ... referenziert. Mit dem Befehl return kann man die Funktion verlassen und einen selbst festgelegten Rückgabewert übergeben. Damit ist es möglich, Ergebnisse der Funktion an die aufrufende Stelle zurück zu geben. Alternativ kann ein Resultat auch per globale Variable oder per echo-Befehl an den Aufrufer zurückgeliefert werden.
In einer Funktion können Skript-interne Variablen (z.B. LOCVAR="value") und globale Variablen (z.B. export GLOBVAR="value") manipuliert werden. Eine globale Variable im Skript, die in der Funktion verändert wird, hat nach Verlassen der Funktion den Wert, den sie in der Funktion erhalten hat. Innerhalb von Funktion sind lokale Variablen (local LOCFUNCVAR="value") möglich.

Da das Thema Shell-Programmierung zum üblichen Curriculum akademischer Einrichtungen im Informatik-Bereich gehört, findet man auch zahlreiche kurze und gute Anleitungen im Netz, die als Orientierungshilfe dienen können. Zusätzlich steht für die bash eine gute Einführung (Bash Beginners Guide) von M. Garrels zur Verfügung.

Tim Wächter
tim.waechter@fh-bielefeld.de