Übungsblatt 4 Lösungsbeispiel Fakultät Elektronik und Informatik
Übungen zu Betriebssysteme Prof. Dr. Rainer Werthebach

Hinweis: Die Lösungen sind teilweise in Form von Kommentaren
(Zeilen beginnend mit #) erklärt.

Aufgabe 1

Erstellen Sie ein Skript, dem Sie einen Dateinamen als Parameter übergeben können. Falls die Datei existiert und lesbar ist, bietet das Skript eine Auswahl an Aktionen zur angegebenen Datei an. Im einzelnen soll es den Dateiinhalt, den Dateityp, die Dateieigenschaften oder die Dateigröße anzeigen können (siehe `file, stat, du`).

```
#!/bin/bash
#falls als erster Parameter "$1" keine(!) lesbare Datei "-r"
#angegeben wurde
if [ ! -r $1 ]
then echo "Keine lesbare Datei angegeben"; exit;
fi
echo "Die Datei $1 existiert und ist lesbar."
echo "1 um den Dateiinhalt auszugeben,"
echo "2 um den Dateityp anzuzeigen,"
echo "3 um die Dateieigenschaften anzuzeigen,"
echo "4 um die Dateigroesse anzuzeigen."
echo -n "Auswahl: "
#"-n" verhindert den Zeilenumbruch nach der Ausgabe
#einlesen in die Variable "auswahl"
read auswahl
#überprüfen, ob der Inhalt von "auswahl" mit 1, 2, 3 oder 4
#beginnt, und die entsprechende Befehlszeile ausführen
case $auswahl in
     1* ) echo "****** ANFANG"; cat $1; echo -e "\nENDE *****" ;;
     2* ) file $1 ;;
     3* ) stat $1 ;;
     4* ) du -h $1 ;;
     *  ) echo $auswahl ist keine gültige Option ;;
esac
```

Die Ausdrücke `"1*"`, `"2*"`, … stehen für beliebige Eingaben die jeweils mit `"1"`, `"2"`, … beginnen. (Ohne die "*“ würde auf exakte Übereinstimmung geprüft werden, "1“ wäre also eine gültige Eingabe, "1x“ nicht)

Das abschließende "*“ matcht alles, was nicht schon vorher gematcht worden ist (entspricht "default“ bei C)

Aufgabe 2

(a) Schreiben Sie ein Skript, welches überprüft, ob genau 2 Parameter angegeben wurden. Sind es weniger oder mehr als 2 Parameter soll eine entsprechende Fehlermeldung ausgegeben werden. (z.B.: „Fehler: zu viele Parameter.“)

(b) Erweitern Sie Ihr Skript, so dass bei genau 2 Parametern überprüft und ausgegeben wird, ob es sich bei den Parametern (oder einem davon) um eine Datei oder ein Verzeichnis im HOME-Verzeichnis handelt. Wenn ja, soll eine entsprechende Meldung ausgegeben werden.
(z.B. „`readme` existiert in /home/94711“)

```
#!/bin/bash

#Anzahl der Parameter($#) kleiner (less than) 2?
if [ $# -lt 2 ]
then
     echo "Fehler: Zu wenige Parameter"
     exit 1

#Anzahl der Parameter($#) größer (greater than) 2?
elif [ $# -gt 2 ]
then
     echo "Fehler: Zu viele Paramter"
     exit 1
else

     #$HOME/$1 ist Datei ODER Verzeichnis?
     if [ -f $HOME/$1 -o -d $HOME/$1 ]
     then
            echo "$1 existiert in $HOME"
     fi

     #$HOME/$2 ist Datei ODER Verzeichnis?
     if [ -f $HOME/$2 -o -d $HOME/$2 ]
     then
            echo "$2 existiert in $HOME"
     fi
fi
```

Aufgabe 3

Schreiben Sie ein kleines Ratespiel, bei dem der Benutzer eine Zahl zwischen 0 und 10 erraten soll. Er hat hierfür unendlich viele Versuche. Das Skript soll ihn nach Fehlversuchen unterstützen, indem es eine Ausgabe mit „Geheime Zahl ist größer“ oder „Geheime Zahl ist kleiner“ macht. Wenn der Benutzer richtig getippt hat, wird ihm gratuliert und das Skript beendet sich.

```
#!/bin/bash
#$RANDOM ist eine Umgebungsvariable die eine Zufallszahl enthält
#% 11 ist eine Modulooperation mit Ergebnis zwischen 0 und 10
zahl=$(expr $RANDOM % 11)
echo "Raten Sie eine Zahl zwischen 0 und 10"
while [ 1 -eq 1 ] # Endlosschleife
do
     echo -n "Ihr Tipp: "
     read input
     #-eq für equals (numerischer Vergleich auf Gleichheit)
     if [ $input -eq $zahl ]
     then
          echo "Die Zahl ist richtig!"
          break
     #-lt für less then ($input ist kleiner als $zahl)
     elif [ $input -lt $zahl ]
     then
          echo "Die Zahl ist zu klein."
     elif [ $input -gt $zahl ]
     then
          echo "Die Zahl ist zu groß."
     fi
done
```

Aufgabe 4

Erstellen Sie eine Shellfunktion add(), welche zwei Zahlen addiert und die Rechnung sowie die Lösung (z.B. „5 + 3 = 8“) auf dem Bildschirm ausgibt (siehe man expr).

So sieht eine mögliche Lösung aus:

```
     add() {
          sum=$(expr $1 + $2)
          echo "$1 + $2 = $sum"
     }
bzw.
     function add {
          sum=$(expr $1 + $2)
          echo "$1 + $2 = $sum"
     }
```

Der Befehl „`expr`“ wertet Ausdrücke, hier eine einfache Addition, aus. Er wird innerhalb einer Subshell aufgerufen und das Berechnungsergebnis wird in der Variable „`sum`“ gespeichert und anschließen ausgegeben.

Achtung: Die Variablen „`$1`“ bis „`$9`“ sind hier nicht dieselben, die eurem Skript als Kommandozeilenparamter übergeben werden, sondern sind die Parameter für die Funktion.
Die Funktion wird also innerhalb eines Skriptes wie ein normales Kommando aufgerufen:

```
     add 5 7
```

Hier entspräche „`$1`“ der „`5`“ und „`$2`“ der „`7`“.