– Lösung zur Praktikumsaufgabe 3 –

Thema: *Shellskripte*

2. a) Die Lösung demonstriert beide Formen des `test`-Statements. Es ist stets eine gute Idee, so viel wie möglich Aspekte der Kommanozeilenparameter zu testen. Man glaubt gar nicht, auf welche Ideen die Nutzer so kommen.

```
#!/bin/bash

# no parameters -> exit
if [ $# -lt 2 ]
then
    echo Usage: $0 \<pattern\> \<file\>
    exit 1
fi

# too many parameters -> warning (no exit)
if test $BASH_ARGC -gt 2
then
    echo $0: Ignoring parameters after $2
fi

# main
echo 'Tiere in der Bibel'
echo  ------------------
echo $1: `grep -c $1 $2`

exit 0
```

   b) Folgender Test kommt hinzu:

```
# test, whether <file> exists
if [ ! -f $2 ]; then
    echo $0: Could not find $2. Stop.
    exit 2
fi
```

3. Die Hauptschleife des Programms könnte beispielsweise so aussehen:

```
read tier
while [ "$tier" != "" ]
do
  echo `grep -o $tier $1 | wc -l` $tier
  read tier
done
```

Anmerkungen:

- Verwenden Sie keine zusätzlichen Ausgaben zur „Menüführung“ des Nutzers, diese kommen Ihnen dann bei der Umleitung der Standardeingabe ins Gehege!
- Kleinschreibung ist Standard bei Variablenbezeichnern, um sie von Umgebungsvariablen, die in Großbuchstaben notiert werden, zu unterscheiden. Dies ist eine

Konvention, kein Dogma.

- Um einen Ausdruck im Skript auszuwerten, wird er in Hochkommata (‘) geklammert. Die Shell substituiert automatisch das entsprechende Ergebnis.
- Die Shell kennt nur synchrones Warten (bei `read`). Actionspiele sind also ohne externe Hilfe schwierig zu programmieren.
- Um später schön nach der Häufigkeit sortieren zu können, geben wir zunächst die Anzahl, gefolgt vom Suchbegriff (Tier) aus.
- `grep` kann zwar mit der Option `-c` auch selbst zählen, dadurch gehen aber doppelte Vorkommen des Suchbegriffs verloren. Besser ist es also, *jedes* Auftreten des Begriffes auszugeben und dann extern die Zeilen zu zählen.

Die Datei mit Suchbegriffen (jeweils einer pro Zeile) heiße `gesuchte-tiere.txt`. Dann leistet der folgende Aufruf das gewünschte:

```
~> ./aufgabe-03-03.sh bibel.txt < gesuchte-tiere.txt | sort -n -r
```

4.* Hauptproblem dürfte die fehlene Funktion zur direkten Längenbestimmung einer Zeichenkette sein. Ein Behelf ist die Ausgabe jedes Dateinamens mittels `echo` und anschließende Zählung der ausgebenen Bytes (`wc`). Genauso kann die Funktion in einem einzeiligen `awk`-Programm realisiert werden (vgl. Aufgabenstellung zum 4.Praktikum). Ein wenig problematisch ist noch die Nutzung numerischer Variablen, die eigentlich erst im 5. Praktikum (und dort nur fakultativ) behandelt wird.

```
#!/bin/bash

# no parameters -> exit
if [ $# -lt 1 ]
then
    echo Usage: $0 \<file\>
    exit 1
fi

# test, whether <dir> exists and is a directory
if [ ! -d $1 ]
then
    echo $0: $1 does not exist or is not a directory. Stop.
    exit 2
fi

# main
max=0
cd $1
for file in *
do
    let length=`echo $file | wc -c`-1
    if [ $max -lt $length ]
    then
  let max=length
```

```
    fi
done
echo $max

# epilogue
exit 0
```

Die Lösung demonstriert weiterhin die Nutzung der `for`-Schleife zur Arbeit mit Dateinamen.

Interessanterweise gibt es für das `wc`-Kommando den Switch `-L`, der das Maximum der Länge aller übergebenen Zeilen ausgibt. Zusammen mit Switch `-1`, der `ls` anweist, alle Ausgaben zeilenweise vorzunehmen, können wir damit obiges Programm in einen handlichen Einzeiler überführen:

```
#!/bin/sh
ls -1 | wc -L
```