Unix-Datenbanken I
SS 2001
10. Juli 2001
Beispiellösung

Bearbeitungszeit: 120 Minuten

Name:	
Vorname:	
Matrikelnummer:	
Studiengang:	

Aufgabe	Punkte	Bewertung
1	**6**	
2	**6**	
3	**10**	
4	**12**	
5	**10**	
6	**8**	
7	**14**	
8	**14**	
Summe:	**80**	

Bitte benutzen Sie für die Lösungen den freigelassenen Platz nach der jeweiligen Angabe; sollte dieser nicht ausreichen, verwenden Sie bitte die Rückseite, wobei die Zuordnung zur jeweiligen Aufgabe deutlich erkennbar sein muss!

Bitte nicht mit Bleistift schreiben – ausgenommen die Skizze in Aufgabe 8!
Die Angabe besteht aus 11 Seiten (ohne Titelblatt)!
Viel Erfolg!!!

Aufgabe 1 **(6 Punkte)**

Schreiben Sie ein kleines *bash*-Skript, das als Argumente einen oder mehrere *login*-Namen bekommt und an die Standardausgabe mitteilt, welcher davon an dem Rechner, wo dieses Skript ausgeführt wird, angemeldet oder auch nicht angemeldet ist. Es sollen nur diese Informationen und keine weiteren Ausgaben erscheinen – ausgenommen die übliche *usage*-Meldung!

Hinweis: Das Kommando *who* liefert, wer angemeldet ist – in der ersten Spalte (Trenner sind *blank*s) erscheint der *login* Name, der nach Konvention kein *blank* enthalten darf!

Lösung:

```
if [ $# -eq 0 ]
then
  echo "Usage: $0 user ... "
  exit 1
fi

while [ $# -ne 0 ]
do
   if who | egrep "^$1 " > /dev/null
   then
     echo "$1 is logged on"
   else
     echo "$1 is not logged on"
   fi
   shift
done
```

Aufgabe 2 **(6 Punkte)**

Das Kommando **date** liefert mit dem Argument **+%H** die aktuelle Tagesstunde (eine Zahl also zwischen 0 und 23). Schreiben Sie ein kleines *bash*-Skript, das abhängig von der aktuellen Stunde eine Gruss-Meldung ausgibt; zwischen 0 und 11 Uhr ein "Guten Morgen", zwischen 12 und 17 Uhr ein "Guten Tag" und danach ein "Guten Abend".

Lösung:

```
std=`date +%H`

case "$std"
in
   0? | 1[01] ) echo "Guten Morgen" ;;
   1[2-7]     ) echo "Guten Tag" ;;
   *          ) echo "Guten Abend" ;;
esac
```

Aufgabe 3 **(10 Punkte)**

Schreiben Sie ein *bash*-Skript zum sog. $3a + 1$ Problem; es erhält seine Eingabe als Argument über die Kommandozeile.
Bei dieser Eingabe muss es sich um (genau) eine natürliche Zahl handeln. Diese wird, falls sie gerade ist, halbiert, und sonst „mal 3 plus 1“ genommen, bis man bei der Eins landet. Der jeweils errechnete Wert soll an die Standardausgabe geschrieben werden.
Ein Beispiel:

```
melzer@thales:~/> aplus1 3
3 10 5 16 8 4 2 1
```

Auf entsprechende *usage*-Meldungen bei Fehlgebrauch ist ebenso wie auf entsprechende *exit*-Stati zu achten!
Lösung:

```
if [ $# -lt 1 ] ; then echo "Usage: $0 natZahl" ; exit 1  ; fi
a=$1
if [ $a -lt 1 ] ; then echo "Usage: $0 natZahl" ; exit 2  ; fi

while [ $a -gt 1 ]
do
  echo -n "$a "
  if [ `expr $a % 2` -eq 0 ] ; then
     a=`expr $a \/ 2`
  else
   a=`expr $a \* 3 + 1`
  fi
done
echo $a
```

Alternativ:

```
if [ $# -lt 1 ] ; then echo "Usage: $0 natZahl" ; exit ; fi
if [ $1 -lt 1 ] ; then echo "Usage: $0 natZahl" ; exit ; fi
echo -n "$1 "
if [ $1 -lt 2 ] ; then echo ; exit ; fi
if [ `expr $1 \/ 2 \* 2` -eq $1 ] ; then
   3ap1 `expr $1 \/ 2`
else
   3ap1 `expr $1 \* 3 + 1`
fi
```

Aufgabe 4 **(12 Punkte)**

Schreiben Sie ein bash-Skript, das aus einem **makefile** für C-Programme die Namen der Quelldateien herausfiltert und diese auf einer Zeile an die Standardausgabe ausgibt. Sie können davon ausgehen, das in diesem *makefile* keine Makros verwendet werden. Beispiel:

```
# MAKEFILE

main:    main.o eingabe.o ausgabe.o
    gcc -Wall -o main main.o eingabe.o ausgabe.o

main.o:     main.c main.h eingabe.h ausgabe.h
     gcc -Wall -c main.c

eingabe.o:    eingabe.c
     gcc -Wall -c eingabe.c

ausgabe.o:    ausgabe.c
     gcc -Wall -c ausgabe.c

clean:
     rm main.o ausgabe.o eingabe.o
```

- Sie dürfen **keine** UNIX-Kommandos, sondern nur Shell-*builtins* verwenden!
- Mehrfach vorkommende Dateinamen dürfen entsprechend oft ausgegeben werden.
- Sollten Sie das *builtin* **set** verwenden, beachten Sie bitte, dass das *makefile* auch leere Zeilen enthalten kann und dass *set* ohne Argumente die aktuellen Shell-Variablen ausgibt!
- Der Name des *makefile*s ist das einzige Argument des zu entwickelnden Skripts!

Lösung bitte auf die folgende Seite!

Aufgabe 4 – Lösung:

```
# get C source files from simple makefile

if [ $# -ne 1 ]
then
  echo "Usage: $0 makefile" >&2
fi

file="$1"

while read line
  do
    if [ "$line" == "" ]
    then
       continue
    fi
    set $line
    for word
    do
      case $word
      in
      *.c | *.h  )  echo -n "$word " ;;
      *          )                   ;;
      esac
    done
done < $file
```

Aufgabe 5 **(10 Punkte)**

Gegeben seien zwei Dateien **A** und **B**, in denen durch Doppelpunkt getrennt Datensätze stehen. In beiden Dateien enthält die erste Spalte die Werte eines Attributes **X** – in Datei **A** ist dieses der Primärschlüssel, in Datei **B** ein Fremdschlüssel (beachten Sie, dass hier ein Wert durchaus mehrfach vorkommen kann!).

Die Aussage, dass die Beziehungsintegrität erfüllt ist, kann man mathematisch auch wie folgt formulieren:

- **A** und **B** sind Mengen von Tupeln des kartesischen Produkts ihrer jeweiligen Attributwertebereiche.
- **A.X** resp. **B.X** ist die Projektion, d.h. die Menge der X-Werte in **A** resp. **B**.
- Es muss gelten: $|A.X| = |A.X \cup B.X|$

Formulieren Sie diese Bedingung durch einen Ausdruck, der in eine **if**-Anweisung eingesetzt werden kann. Er soll also wie folgt verwendet werden können:

```
if # IHRE LOESUNG
then
   echo "alles okay mit der Beziehungsintegritaet"
else
   echo "Beziehungsintegritaet ist verletzt"
fi
```

Lösung:

```
if [ `cut -d: -f1 A | wc -l` -eq `cat A B | cut -d: -f1 | sort -
u | wc -l` ]
then
  echo "Okay"
else
  echo "Error"
fi
```

Aufgabe 6 **(8 Punkte)**

Gegeben seien wie in Aufgabe 5 die zwei Dateien **A** und **B**, in denen durch Doppelpunkt getrennt Datensätze stehen. In beiden Dateien enthält die erste Spalte die Werte eines Attributes **X** – in Datei **A** ist dieses der Primärschlüssel, in Datei **B** ein Fremdschlüssel.

Jetzt sollen Sie mit einem ausführbaren *tcl*-Skript herausfinden, welche Datensätze aus **B** die Beziehungsintegrität verletzen (diese sind auszugeben!).

Lösung:

```
#!/bin/sh
# the next line restarts using tclsh \
exec tclsh "$0" "$@"

set a [open "A" r ]
set b [open "B" r ]

while { [gets $a line ] >= 0 } {
    set line [ split $line : ]
    set feld([lindex $line 0]) 1
}

while { [gets $b line ]  >= 0 } {
    set line [ split $line : ]
    if { ![ info exists feld([lindex $line 0]) ] } {
       puts "Wrong: $line"
    }
}

close $a
close $b
```

Aufgabe 7 **(14 Punkte)**

Gegeben sei folgender *screenshot*:

Geben Sie das zugehörige *wish*-Programm an (nur Definitionen und *pack*-Anweisungen, keine Funktionalität und keine Eigenschaften, die das Verändern betreffen)!
Lösung:

```
#!/bin/sh
#the next line restarts using wsh \
exec wish "$0" "$@"

wm title . "Klausuraufgabe"

frame .left
frame .right
frame .right.up
frame .right.down
frame .right.up.oben
frame .right.up.unten

button .right.up.oben.eins -text "Eins" -anchor e
button .right.up.oben.zwei -text "Zweimal"
button .right.up.unten.drei -text "Drei"
button .right.up.unten.vier -text "Vier"
pack .right.up.oben.eins -fill x -side top
pack .right.up.oben.zwei -fill x -side top
pack .right.up.unten.drei -fill x -side top
pack .right.up.unten.vier -fill x -side top
pack .right.up.oben .right.up.unten -fill x -side left
```

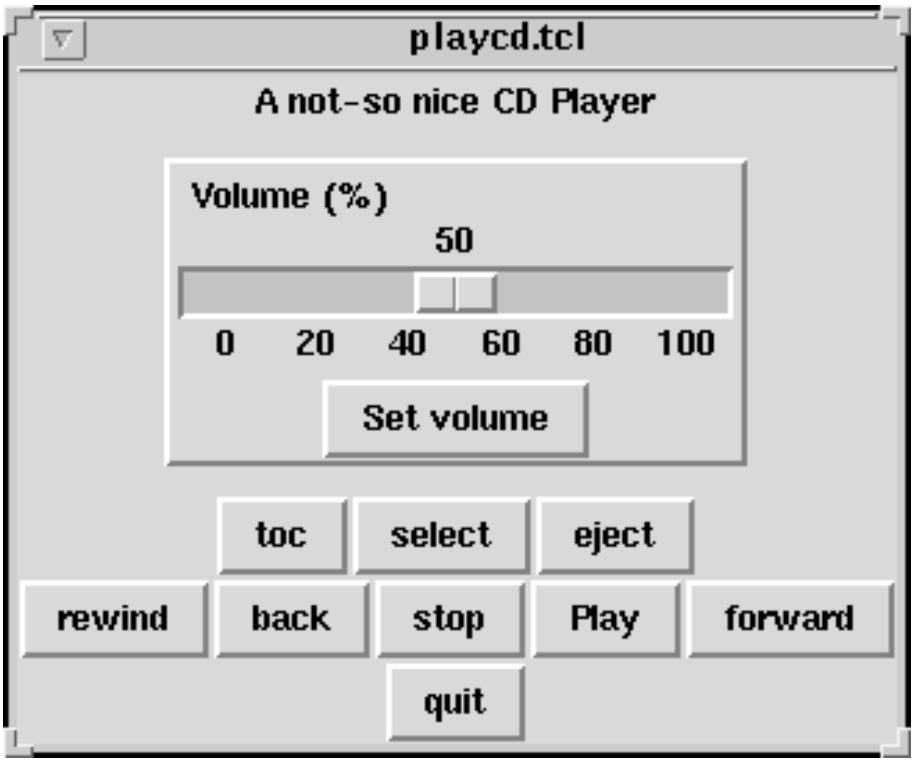
```
label .right.down.eins -text "Eine Textzeile"
label .right.down.zwei -text "Und noch eine Zeile"
pack .right.down.eins .right.down.zwei -side top

pack .right.up .right.down -side top

label .left.eins -text "One"
label .left.zwei -text "Two"
label .left.drei -text "Three"
pack .left.eins .left.zwei .left.drei -side top

pack .right .left -side left
```

Aufgabe 8 **(14 Punkte)**

Gegeben sei folgendes *wish*-Programm:

```
#!/usr/local/bin/wish

proc stopit { } {
   global play
   set play "Play"
}

proc playstop { } {
   global play
   if {$play == "Play"} {
      set play "Pause"
   } elseif {$play == "Pause" } {
      set play "Unpause"
   } else {
      set play "Pause"
   }
}

proc sel { } {
   global titles acttitle
   toplevel .selectwin
   after 100 { grab -global .selectwin }
   scale .selectwin.scale -from 1 -to $titles -variable acttitle\
      -tickinterval 1 -length [expr $titles * 20]\
      -showvalue true -orient horizontal -label "Select Title"
   button .selectwin.quit -text OK \
      -command { destroy .selectwin }
   pack .selectwin.scale .selectwin.quit -side top
}

set titles 20
set acttitle 1
set play "Play"
set volume 50

frame .main -width 400 -height 200
pack .main
label .main.text -text "A not-so nice CD Player"

frame .main.vol -relief raised -borderwidth 2
scale .main.vol.vol -from 0 -to 100 -length 200 -tickinterval 20 \
   -label "Volume (%)" -orient horizontal -variable volume
```

```
button .main.vol.set -text "Set volume" -command {exec xset b $volume}
pack .main.vol.vol .main.vol.set -side top -expand true

frame .main.buttons
button .rewind -text "rewind" -command {puts "rewind" }
button .back -text "back" -command {puts "back" }
button .stop -text "stop" -command stopit
button .play -textvar play -command playstop
button .forward -text "forward" -command {puts "forward" }
button .main.exit -text quit -command exit
pack .rewind .back .stop .play .forward -in .main.buttons -side left

frame .main.aps
button .toc -text "toc" -command {puts "toc" }
button .select -text "select" -command sel
button .eject -text "eject" -command {puts "eject" }
pack .toc .select .eject -in .main.aps -side left
pack .main.text -side top
pack .main.vol -pady 10 -side top -expand true
pack .main.aps .main.buttons .main.exit -side top
```

Geben Sie auf der nächsten Seite in einer guten Skizze (inkl. der Beschriftungen) an, mit welcher Oberfläche dieses Programm startet!

Aufgabe 8 – Lösung: