Anhang A

Lösungen der Aufgaben

Einführung

In diesem Kapitel des Anhangs sind Lösungsvorschläge zu den Aufgaben der Shell-, LaTeX- und Programmier-Kapitel abgedruckt. Wenn Ihre Lösung nicht genau so aussieht, heißt das aber nicht, dass diese unbedingt falsch sein muss. Gerade bei der Programmierung und auch bei LaTeX führen viele Wege zum Ziel.

A.1 Shell

1. Es geht um Kommandosubstitution. Was beinhaltet nach Abarbeitung der Kommandofolge die Datei `datei3`?

```
echo "blau" > kunde
echo "gruen" > kun.de
echo "gelb" >> kunde
echo "rot" > kunde.r
echo kund* >datei2
cat ‘cat datei2‘ > datei3
```

Lösung

```
echo "blau" > kunde         # blau

echo "gruen" > kun.de       # gruen

echo "gelb" >> kunde        # blau
                            # gelb

echo "rot" > kunde.r        # rot

echo kund* >datei2          # kunde
                            # kunde.r

cat ‘cat datei2‘ > datei3   # blau
                            # gelb
                            # rot
```

2. Mit Pipes kann man recht komfortable Kommandos bilden. Verwenden Sie die Hintereinanderschaltung von `ps aux` und `grep`, um die Prozesse eine bestimmten Benutzers (z. B. root) oder eines bestimmten Terminals anzuzeigen.

Lösung

```
# zum Beispiel:
user$ ps aux | grep sshd
root   316  0.0  0.3  2896 1300 ?   Ss  09:23  0:00 /usr/sbin/sshd
root   453  0.0  0.4  5912 1768 ?   S   09:36  0:00 /usr/sbin/sshd
plate  455  0.0  0.5  5920 1880 ?   S   09:36  0:00 /usr/sbin/sshdpe
```

3. Geben Sie einen Aufruf von `find` an, der im Verzeichnis `/home` alle Dateien sucht, auf die innerhalb der letzten 10 Tage zugegriffen wurde.

Lösung

```
find /home -mtime -10 2>/dev/null
# die Umleitung der Fehlerausgabe beseitigt Fehlermeldungen,
# wenn man für das entsprechende Verzeichnis keine
# Zugriffsrechte besitzt
```

4. Welches `find`-Kommando müsste man verwenden, um in allen Dateien auf der Platte mit der Endung „.txt“ nach der Zeichenkette „UNIX“ zu suchen.

Lösung

```
find / -name ’*.txt’ | xargs grep "UNIX"
```

5. Schreiben Sie ein Shell-Skript, das in einer Endlosschleife alle Sekunden die Uhrzeit (Stunde, Minute und Sekunde) in der linken oberen Ecke des Bildschirms ausgibt.

Lösung

```
while :
  do
  tput home
  date "+[%H:%M:%S]"
  sleep 1
  done
```

6. Schreiben Sie ein Shell-Skript „ggt“, daß den größten gemeinsamen Teiler der als Parameter übergebenen beiden Zahlen berechnet. Formulieren Sie die Berechnung des GGT als Shell-Funktion. Der Algorithmus lautet folgendermaßen:

```
ggt(x,y):
solange x ungleich y ist, wiederhole
  falls x > y  dann  x = x - y
  sonst              y = y - x;
```

Denken Sie daran, daß man zum Rechnen das Kommando `expr` braucht.

Lösung

```
x=$1
y=$2
while [ $x -ne $y ]
  do
  if [ $x -gt $y ]; then
    x=‘expr $x - $y‘
  else
    y=‘expr $y - $x‘
  fi
  done
echo $x
```

7. Eine Datei namens `personal` habe folgendes Aussehen:

```
[Name]         [Vorname]     [Wohnort]      [Geb.-Datum]
=======================================================
Meyer          Peter         Berlin          10.10.1970
Schulze        Axel          Hamburg         12.12.1980
Lehmann        Rita          München         17.04.1971
```

Sortieren Sie die Daten der Datei (ohne Zeile 1 und 2) nach dem Namen und geben Sie das Ergebnis in eine Datei `personal.sort` aus. Die beiden Überschriftszeilen sollen natürlich wieder am Dateianfang stehen.

Lösung

```
head -2 personal > hilf
cat personal | sed -e '1,2d' | sort >> hilf
mv hilf personal
```

8. Schreiben Sie ein Shellscript, das an alle Benutzer mit der Gruppennummer 100 eine E-Mail verschickt. Betreff und die Datei, welche den E-Mailtext enthält, werden als Parameter an das Script übergeben.

Lösung

```
Subject=$1
File=$2
# Hier waere eine Fehlerbehandlung ganz nett
if [ ! -f "$File" ]; then
  echo "File $File not found/not readable"
  exit 1
  fi

if [ "$Subject" == "" ]; then
  Subject="Automatic Mail from Sysadmin"
  fi

# /etc/passwd zeilenweise bearbeiten
while read LINE
  do
  # Usernamen und Gruppennummer aus der Zeile fischen
  USR=`echo $LINE | cut -f1 -d:`
  GID=`echo $LINE | cut -f4 -d:`
  if [ $GID -eq 100 ]; then         # wenn die Gruppe passt
    echo "Sending $File to $USR"    # Mail verschicken
    mailx -s "$Subject" < $File
  fi
  done </etc/passwd
```

A.2 LaTeX

1. Schreiben Sie eine Muster-Präambel für einem Artikel, den Sie mit LaTeX verfassen wollen.

Lösung

```
\documentclass{article} % keine Kapitel
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[ngerman]{babel} % Neue Rechtschreibung
\usepackage{graphicx}
\begin{document} % Hier geht der Text los

\title{Ein toller Titel}
\author{Da stehe ich}
\maketitle
\newpage

\tableofcontents
\newpage
Hier geht dann der Text los
```

2. Erstellen Sie den LaTeX-Sourcecode für folgende Auflistung:

   Checkliste für mehrtägige Exkursionen

   - Waschzeug
     - Handtuch
     - Zahnbürste
     - Zahnpasta
     - Kamm/Bürste
     - Seife/Duschgel
   - Hausschuhe
   - Kleidung (genug)
   - Unterwäsche (ebenfalls genug)
   - Badehose
   - Trinkflasche
   - Schlafanzug
   - Ohrstöpsel
   - Wärmflasche (für Weicheier)
   - Notfallausrüstung
     - Handy
     - Verbandszeug
     - Alka Selzer
     - Rettungsdecke
     - MP3-Player
     - Taschenmesser

Lösung

```
\textbf{Checkliste für mehrtägige Exkursionen}

\begin{itemize}
  \item Waschzeug
    \begin{itemize}
     \item Handtuch
     \item Zahnbürste
     \item Zahnpasta
     \item Kamm/Bürste
     \item Seife/Duschgel
    \end{itemize}
  \item Hausschuhe
  \item Kleidung (genug)
  \item Unterwäsche (ebenfalls genug)
  \item Badehose
  \item Trinkflasche
  \item Schlafanzug
  \item Ohrstöpsel
  \item Wärmflasche (für Weicheier)
  \item Notfallausrüstung
    \begin{itemize}
     \item Handy
     \item Verbandszeug
     \item Alka Selzer
     \item Rettungsdecke
     \item MP3-Player
     \item Taschenmesser
    \end{itemize}
 \end{itemize}
```

3. Setzen Sie folgende Tabelle mit LaTeX:

Nahrungsmittel	Art	Ausgangsprodukt
Schnitzel	Fleisch	Schwein
Steak	Fleisch	Rind
Semmel	Getreide	Weizen
Pop Corn	Getreide	Mais
Joghurt	Milchprodukt	Kuhmilch
Knödel	Gemüse	Kartoffel

Lösung

```
\begin{tabular}{p{4cm} p{3cm} p{4cm} }
\hline
Nahrungsmittel & Art & Ausgangsprodukt \\
\hline
Schnitzel & Fleisch & Schwein \\
Steak & Fleisch & Rind \\
Semmel & Getreide & Weizen \\
Pop Corn & Getreide & Mais \\
Joghurt & Milchprodukt & Kuhmilch \\
Knödel & Gemüse & Kartoffel \\
\hline
\end{tabular}
```

4. Setzen Sie folgende Formel mit LaTeX:

   Mit $k = \sqrt[3]{8}$:

   $$k^2 \in \left\{ \sum_{i=1}^{\infty} \frac{1}{2^i}, \ldots, \sum_{i=1}^{\infty} \frac{1}{i} \right\}$$

   Lösung

   ```
   Mit \(k=\sqrt[3]{8}\):
   \[ k^2\in\left\{\sum_{i=1}^{\infty}\frac{1}{2^i}, \ldots,
   \sum_{i=1}^{\infty}\frac{1}{i}\right\} \]
   ```

5. Versuchen Sie, folgende kleine Mathelektion zu setzen:

   Jeder Mathematiker weiß, dass z. B. die Summe von zwei Größen nicht etwa in der Form

   $$1 + 1 = 2 \tag{A.1}$$

   dargestellt wird. Diese Form ist viel zu schlicht. Schon Anfangssemester wissen, dass gilt:

   $$1 = \ln e \tag{A.2}$$

   weiterhin ist geläufig, dass

   $$1 = \sin^2 q + \cos^2 q \ . \tag{A.3}$$

   Ausserdem ist dem kundigen Leser offensichtlich, dass

   $$2 = \sum_{n=0}^{\infty} \frac{1}{2^n} \ . \tag{A.4}$$

Daher kann die Gleichung (A.1) viel wissenschaftlicher in der Form

$$\ln e + (\sin^2 q + \cos^2 q) = \sum_{n=0}^{\infty} \frac{1}{2^n} \ . \tag{A.5}$$

ausgedrückt werden.

Lösung

```
Jeder Mathematiker weiß, dass z.\,B.\ die Summe von zwei Größen nicht
etwa in der Form
\begin{equation}\label{eq:simple}
1 + 1 = 2
\end{equation}
dargestellt wird. Diese Form ist viel zu schlicht. Schon Erstsemester
wissen, dass gilt:
\begin{equation}
1 = \ln e
\end{equation}
weiterhin ist geläufig, dass
\begin{equation}
1 = \sin^2 q + \cos^2 q \ .
\end{equation}
Ausserdem ist dem kundigen Leser offensichtlich, dass
\begin{equation}
2 = \sum^{\infty}_{n=0} \frac{1}{2^n} \ .
\end{equation}
Daher kann die Gleichung (\ref{eq:simple}) viel wissenschaftlicher
in der Form
\begin{equation}
\ln e + (\sin^2 q + \cos^2 q) = \sum^{\infty}_{n=0} \frac{1}{2^n} \ .
\end{equation}
ausgedrückt werden.
```

A.3 Programmieren mit Perl

1. Schreiben Sie ein Perl-Programm, das Zahlen einliest und sie aufsummiert. Wird statt einer Zahl der Buchstabe „e“ eingegeben, berechnet das Programm den Mittelwert der Zahlen und gibt ihn aus.

Lösung

```
use strict;

my $sum = 0;
my $count = 0;
# read first number
while (1)
  {
  print "Zahl eingeben: ";
  my $number = <>;
  chomp($number);
  last if ($number eq 'e');
  $count++;
  $sum += $number;
  }
print "$count Zahlen wurden eingegeben\n";
if ($count > 0)
  {
  print "Mittelwert: ",$sum/$count,"\n";
  }
```

2. Schreiben Sie ein Programm, das eine Zahl in Worten ausgibt (etwa für das Bedrucken eines Scheckformulars). Die Zahl soll dem Programm auf der Kommandozeile übergeben werden. Beispiel:

```
user$  perl drucke 1234
-eins-zwei-drei-vier-
user$
```

Verwenden Sie für die Zuordnung zwischen Ziffern und zugehörigem Zahlwort einen Hash.

Lösung

```
use strict;
my %numbers = (0 => "null", 1 => "eins", 2 => "zwei", 3 => "drei",
               4 => "vier", 5 => "fünf", 6 => "sechs", 7 => "sieben",
               8 => "acht", 9 => "neun");

my @characters = split (//, "@ARGV");
print "-";
foreach my $char (@characters)
  {
  $char = $numbers{$char} if ($char ge "0" && $char le "9");
  print "$char-";
  }
print "\n";
```

3. Schreiben Sie ein Programm, das Text von der Standardeingabe einliest und die Häufigkeit der einzelnen Buchstaben ermittelt. Die Liste der Häufigkeiten

soll dann alfabetisch ausgegeben werden. Hinweis: Eine Zeile in Buchstaben zerlegt man mit der Anweisung `@chars = split (//,$line);`.

Lösung

```
my ($char, $line, @chars);
my %freq = ();
while(defined($line = <>))
  {
  chomp $line;
  last if ($line eq '');
  @chars = split (//,$line);
  # Keine Unterscheid. gross/klein
  foreach $char (@chars)
    { $freq{lc($char)}++; }
  }

# Liste alphabetisch ausgeben
foreach $char (sort keys %freq)
  { print "$char: $freq{$char}\n"; }
```

4. Schreiben Sie ein Programm, das beliebig viele Zeilen Text von der Standardeingabe einliest und dien Text in umgekehrter Reihenfolge wieder ausgibt. Dabei sollen die Zeichen innerhalb der Zeile umgekehrt werden (`reverse`) und auch die letzte Zeile als erste ausgegeben werden (`push` und `pop`).

Lösung

```
my @lines = ();
my ($line, @chars);
while(defined($line = <>))
  {
  chomp $line;
  last if ($line eq '');
  # reverse the characters in the line
  # and push this onto a stack
  @chars = split (//,$line);
  @chars = reverse @chars;
  $line = join ("", @chars);
  push @lines, $line;
  }
print "$line\n" while ($line = pop @lines);
```

5. Erzeugen Sie eine Datei namens `gebtag`, die folgenden Text enthält:

```
11.2.1947:Thomas Alva:Edison
4.1.1643:Isaac:Newton
14.3.1879:Albert:Einstein
```

```
27.12.1571:Johannes:Kepler
```

Ergänzen Sie die Datei mit einem Testdatensatz, der das heutige Datum trägt.

Schreiben Sie nun ein Programm das nach der Püfung ob `gebtag` existiert, die Datei einliest und alle Personen heraussucht, die heute Geburtstag haben. Es sollen Vorname, Name und Geburtsjahr ausgegeben werden. Hinweis: Das aktuelle Datum erhalten Sie mit

```
($sek,$min,$std,$tag,$mon,$jahr) = localtime(time);
$mon++; $jahr += 1900;
```

Lösung

```
use strict;

my ($count, $datum, $vorname, $name, $tt, $mm, $jj);
my $datei = "gebtag";
my ($sek,$min,$std,$tag,$mon,$jahr) = localtime(time);
$mon++; $jahr += 1900;

print "Heute ist $tag.$mon.$jahr\n";

$count = 0;
open (GEB, $datei) || die "Kann $datei nicht öffnen: $!\n";
while (<GEB>)
  {
  chomp;
  # Kommentarzeilen erlauben
  next if (/^#/);
  # Zeile auftrennen in Datum Vorname Name
  ($datum,$vorname,$name) = split(/:/,$_);
  # Datum auftrennen in Tag, Monat, Jahr
  ($tt,$mm,$jj) = split(/\./,$datum);
  # Punkt ist spezielles Zeichen in reg. Ausdrücken, daher '\.'

  if ($tt == $tag && $mm == $mon)
    {
    print "$vorname $name hat heute Geburtstag ($jj).\n";
    $count++;
    }
  }
close GEB;
if ($count == 0)
  { print "Heute hat keiner Geburtstag.\n"; }
else
  { print "Heute haben $count Person(en) Geburtstag.\n"; }
```

6. Schreiben Sie ein Programm für den Administrator, das alle Prozesse eine auf der Kommandozeile angegebenen Benutzers killt. Rufen Sie dazu das `ps`-Kommando als Pipe auf und verarbeiten Sie dessen Ausgabe zeilenweise, indem Sie die Prozesse des angegebenen Users heraussuchen. Dann extrahieren Sie die Prozessnummer aus der Zeile. Perl kennt den Befehl `kill(Signal,Prozessnummer)` mit dessen Hilfe Sie die Prozesse löschen können. Führen Sie die Prozedur nacheinander mit den Signalen `TERM`, `HUP` und `KILL` aus.

Lösung (Hauptprogramm)

```
#!/usr/bin/perl
use strict;
# Kommandozeile fuer ps ggf. anpassen:
# Das Kommando liefert Prozessnummer und Username
my $pscommand = '/bin/ps -eo pid,user';

if ($#ARGV != 0)
  { print "Usage: $0 username\n"; exit(1); }

my ($signal, $error);
my $actuser = shift;

foreach $signal ("TERM","HUP","KILL")
  { &killuser($actuser, $signal); }
exit 0;

sub killuser # (user, signal)
  {
  # can only be done as root
  my $user = shift;
  my $sig = shift;
  my @entry = ();
  my $name = '';
  my $pid = 0;
  my $line = '';
  print "Sending $sig signal to all processes ...\n";
  # Prozessliste alle User einlesen, User extrahieren
  open(PS, "$pscommand |");
  @entry = grep(/$user/,<PS>);
  close PS;
  return if ($#entry == -1); # keine Prozesse da
  foreach $line (@entry)
    {
    chomp($line);
    # mehrere Leerzeichen durch eines ersetzen
    $line =~ s/  */ /g;
    # Leerzeichen am Anfang weg
    $line =~ s/^ *//g;
    ' Prozessnummer und Username extrahieren
    ($pid,$name) = split(/ /,$line);
    print "$pid ";
    kill($sig,$pid);
    }
  print "\n";
  return;
  }
```