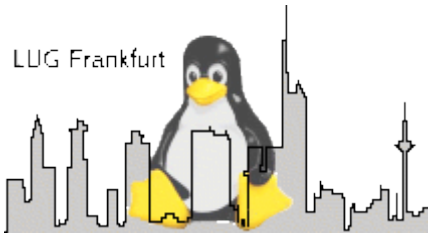
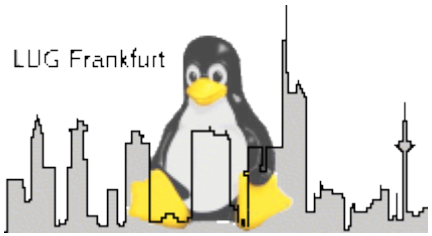
Shell-Programmierung

von Rolf Schmidt für
LUG Frankfurt



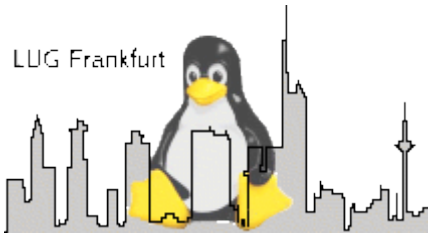
Ziel des Vortrags

- „Skripten“ für Einsteiger
- Arten der Shell-Programmierung
- Möglichkeiten der Shell-Programmierung
- Eigene Skripte erstellen zu können
- Ausgefeilte Programmierung
- Spezialitäten der Bash



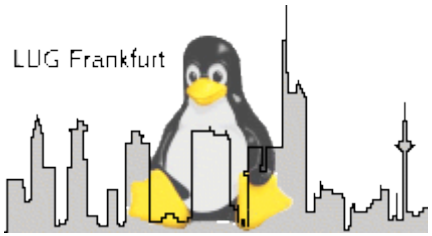
Aufbau des Vortrags

- Verschiedene Shells
- Besonderheiten der BASH
- Besonderheiten der Shell-Programmierung
- Einfache (Batch) Skripte
- Elemente von Programmiersprachen
- Einfache Shell-Skripte
- Komplexere Shell-Skripte
- Fragen und Diskussion



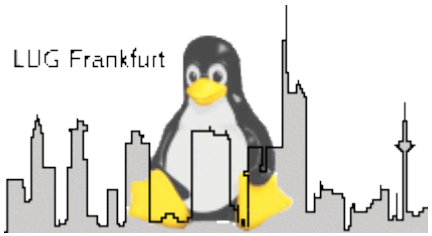
Verschiedene Shells (1)

- Bourne Shell
  - Die erste UNIX-Shell
  - Benannt nach ihrem Progammierer
  - Wird auch die Standard-Shell genannt.
- Korn Shell
  - „History-Mechnismus“
  - Nach ihrem Entwickler benannt
  - Ein kommerzielles Produkt (gibt Public-Domain-Variante)
  - Ist kompatibel zur Bourne-Shell.



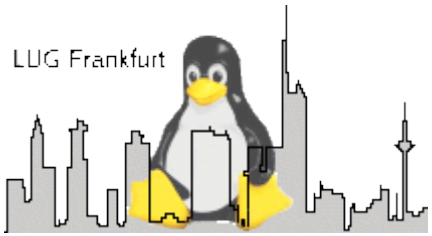
Verschiedene Shells (2)

- Die C-Shell
  - Sprachkonstrukte bzw. Sprachregeln ähnlich der Programmiersprache „C“
  - Möglichkeiten zur Prozesssteuerung wie „bg“ und „fg“
  - „History-Mechanismus“
- Bourne-Again-Shell (BASH)
  - Standard-Shell unter Linux
  - Entwicklung der *Free-Software-Foundation*
  - Umfassendste und komfortabelste Shell



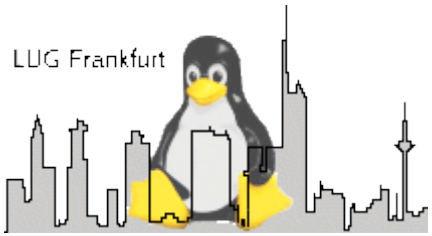
Verschiedene Shells (3)

- A-Shell
  - Eingeschränkter Funktionsumfang
  - Statisch gelinkt
  - Kleines Binary
  - Kann verwendet werden, ohne das ein System vollständig gestartet ist.
  - Häufig in Startumgebungen oder sehr kleinen Systemen.
- Z-Shell...



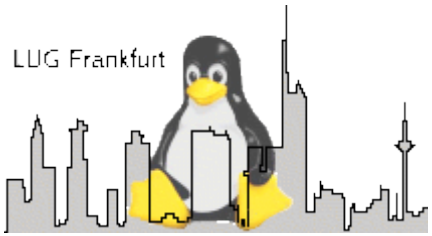
Aufbau des Vortrags

- Verschiedene Shells
- Besonderheiten der BASH
- Besonderheiten der Shell-Programmierung
- Einfache (Batch) Skripte
- Elemente von Programmiersprachen
- Einfache Shell-Skripte
- Komplexere Shell-Skripte
- Fragen und Diskussion



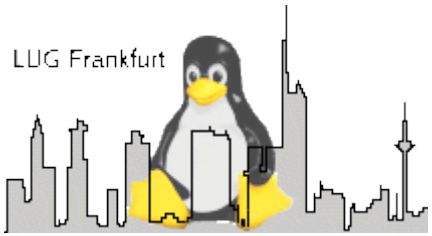
Besonderheiten der BASH

- Abwärtskompatibel mit Bourne-Shell
- „History-Mechanismus“
- Job-Control („bg“ und „fg“-Befehl)
- auch mit Eingeschränkungen startbar
- Frei Software der FSF



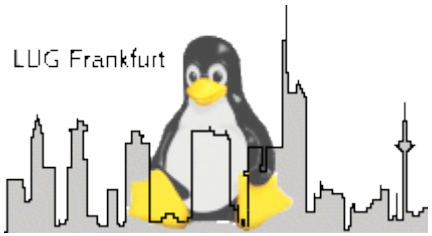
Aufbau des Vortrags

- Verschiedene Shells
- Besonderheiten der BASH
- Besonderheiten der Shell-Programmierung
- Einfache (Batch) Skripte
- Elemente von Programmiersprachen
- Einfache Shell-Skripte
- Komplexere Shell-Skripte
- Fragen und Diskussion



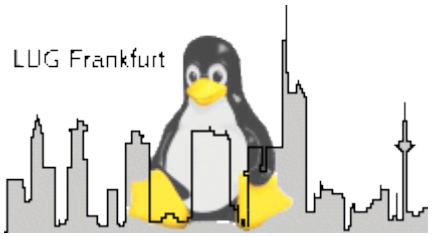
Besonderheiten der Shell-Programmierung

- Ausführen externer Programme als Erweiterung der Fähigkeiten
- Übernahme und Weiterverarbeitung der Ausgabe externer Programme
- Befehlsersetzung `echo $( echo "2 + 3 * 5" | bc )`
- Rückgabewerte auch bei undefinierten oder nicht gesetzten Variablen
- Shebang – (bei allen Skriptsprachen)



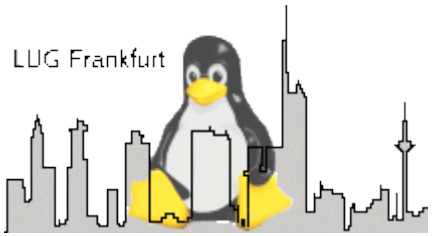
Aufbau des Vortrags

- Verschiedene Shells
- Besonderheiten der BASH
- Besonderheiten der Shell-Programmierung
- Einfache (Batch) Skripte
- Elemente von Programmiersprachen
- Einfache Shell-Skripte
- Komplexere Shell-Skripte
- Fragen und Diskussion



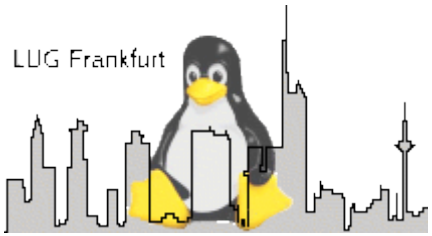
Einfache (Batch) Skripte (1)

```
~$ cat << EOF > rename.sh
cp rename.sh rename1.sh
rm rename.sh
mv rename1.sh rename.sh
EOF
~$ chmod u+x rename.sh
~$ ./rename.sh
```



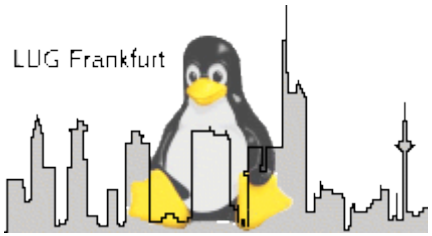
Einfache (Batch) Skripte (2)

```
~$ cat << EOF > rename1.sh
#! /bin/bash
# Mit Aufrufparameter
cp $1 $1.bak
rm $1
mv $1.bak $1
EOF
~$ chmod u+x rename1.sh
~$ ./rename1.sh rename1.sh
```



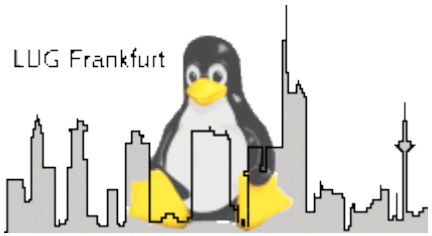
Aufbau des Vortrags

- Verschiedene Shells
- Besonderheiten der BASH
- Besonderheiten der Shell-Programmierung
- Einfache (Batch) Skripte
- Elemente von Programmiersprachen
- Einfache Shell-Skripte
- Komplexere Shell-Skripte
- Fragen und Diskussion



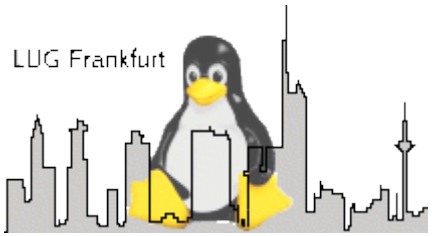
Elemente von Programmiersprachen

1. Umgang mit Variablen
2. Verzweigungen und Fallunterscheidung
3. Schleifen
4. Blockbildung
5. Elementare Mathematik
6. Umgang mit Elementen, die keine Zahlen sind (zum Beispiel Text)
7. Kommentare



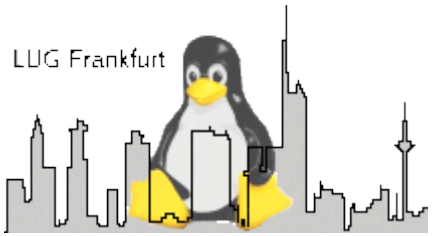
Aufbau des Vortrags

- Verschiedene Shells
- Besonderheiten der BASH
- Besonderheiten der Shell-Programmierung
- Einfache (Batch) Skripte
- Elemente von Programmiersprachen
- Einfache Shell-Skripte
- Komplexere Shell-Skripte
- Fragen und Diskussion



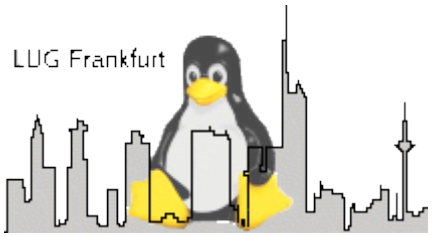
Einfache Shell Skripte

- Shebang
  - #!
- Variablenname vs. Inhalt
  - VAR vs. $VAR
- Rückgabewert
  - 0 kein Fehler, jede andere Zahl = Fehler
- Konventionen
  - Variablennamen in Großbuchstaben



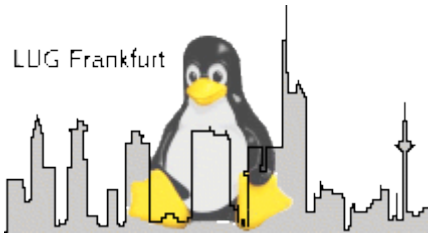
Einfache Shell Skripte

- Schlüsselworte
  - Abfrage:
    - if else fi
    - [ ] resp. `test`
    - testbare Eigenschaften (exist, is link, is executable...)
  - Vergleichsoperatoren
    - == eq
    - != ne
    - > >= gt ge
    - < <= lt le



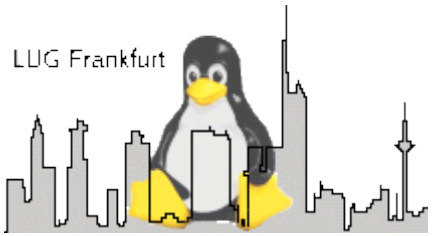
Einfache Shell Skripte (1)

```
~$ cat << EOF > rename2.sh
#! /bin/bash
# einfache Abfrage
if [ $# == 0 ] ; then
  echo Fehlendes Argument
else
  cp $1 $1.bak
  rm $1
  mv $1.bak $1
fi
EOF
~$ chmod u+x rename2.sh
~$ ./rename2.sh
```



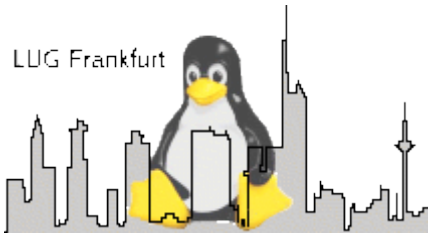
Einfache Shell Skripte

- Spezielle Variablen
  - $? Rückgabewert des vorherigen Programms
  - $0 Der Skriptname
  - $1 - $9 Die mitgegebenen Parameter
    - zusätzliche Parameter müssen mit dem `shift`-Befehl verarbeitet werden
  - $# Anzahl der mitgegebenen Parameter
  - $$ Die Prozess-Nummer des Skripts



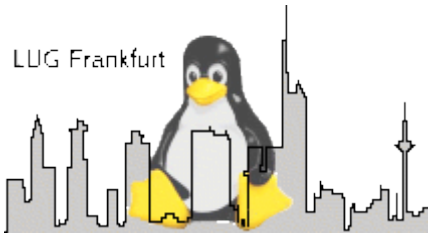
Einfache Shell Skripte (2)

```
~$ cat << EOF > rename3.sh
#! /bin/bash
# Mit eigener Variabel
DATEINAME=$1
if [ $# == 0 ] ; then
  echo Fehlendes Argument
else
  cp $DATEINAME $DATEINAME.bak
  rm $DATEINAME
  mv $DATEINAME.bak $DATEINAME
fi
EOF
~$ chmod u+x rename3.sh
~$ ./rename3.sh
```



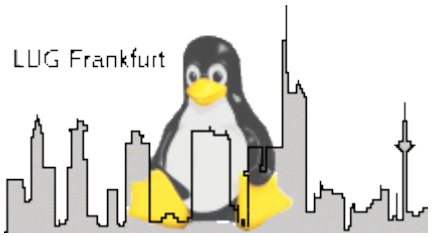
Einfache Shell Skripte

- Schleifen
  - Die „Zählschleife“ mit vorgegebener Anzahl von Durchläufen
  - Die „kopfgesteuerte“ Schleife
  - Die „fußgesteuerte“ Schleife
- Finden Sie die Unterschiede
  - Schreiben Sie dazu „ -x“ hinter die Shebang



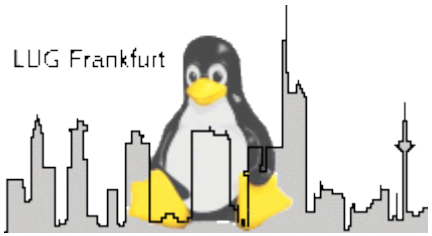
Einfache Shell Skripte (3)

```
~$ cat << EOF > backup.sh
#! /bin/bash
# nützliche Schleife
SICHERN=`ls -1 *.sh`
if [ “$SICHERN“ != ““ ]; then
  mkdir backup
  for i in $SICHERN
  do
    cp $i backup
  done
fi
EOF
~$ chmod u+x backup.sh
~$ ./backup.sh
```



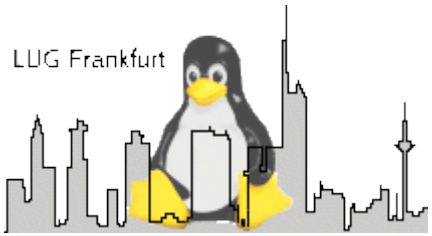
Einfache Shell Skripte (4)

```
~$ cat << EOF > schleife.sh
#! /bin/bash
# einfache (Zähl-)Schleife
for i in 1 2 3
do
  echo $i
done
EOF
~$ chmod u+x schleife.sh
~$ ./schleife.sh
1
2
3
```



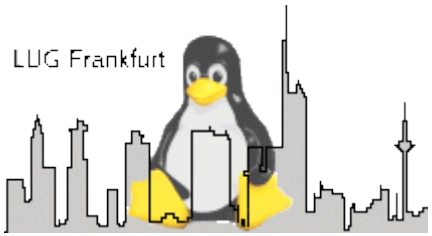
Einfache Shell Skripte (5)

```
~$ cat << EOF > schleife2.sh
#! /bin/bash
# kopfgesteuerte Schleife
while [ “$1“ != “Ende“ ]
do
  echo $1
  shift
done
EOF
~$ chmod u+x schleife2.sh
~$ ./schleife2.sh Dies ist das Ende der Schleife
Dies
ist
das
~$
```



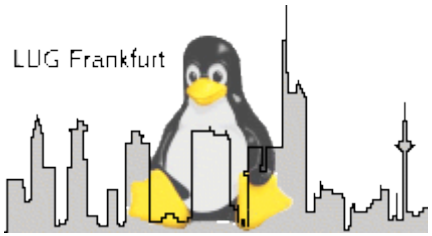
Einfache Shell Skripte (6)

```
~$ cat << EOF > schleife3.sh
#! /bin/bash
# fußgesteuerte Schleife
until [ "$1" = "Ende" ]
do
  echo $1
  shift
done
EOF
~$ chmod u+x schleife3.sh
~$ ./schleife2.sh Dies ist das Ende der Schleife
Dies
ist
das
~$
```



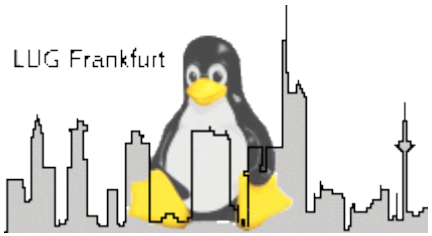
Einfache Shell Skripte

- Benutzerinteraktion
  - Argumente werden durch „white spaces“ aufgetrennt.
  - IFS listet die Trennzeichen
  - Evtl. quoting notwendig
  - Zuweisung von Werten an verschiedene Variable



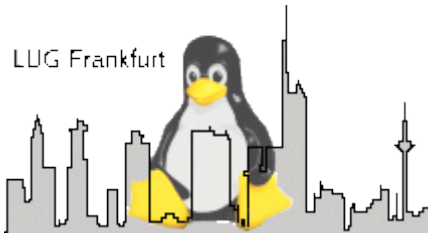
Einfache Shell Skripte (7)

```
~$ cat << EOF > eingabe
#! /bin/bash
IFS=""
read -ea VAR -p "Bitte eine Zeile eingeben: "
echo $VAR
EOF
~$
~$ chmod u+x ./eingabe.sh
./eingabe
Bitte eine Zeile eingeben: Hallo Onkel Fritz.
Hallo Onkel Fritz.
~$
```



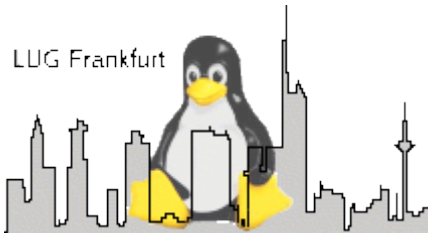
Einfache Shell Skripte (8)

```
~$ cat << EOF > eingabe2.sh
#! /bin/bash
read -ep "Bitte eine Zeile eingeben: " VAR0 VAR1
for i in 0 1
do
  case $i in
    0)
      echo $VAR0
    ;;
    1)
      echo $VAR1
    ;;
  esac
done
EOF
```



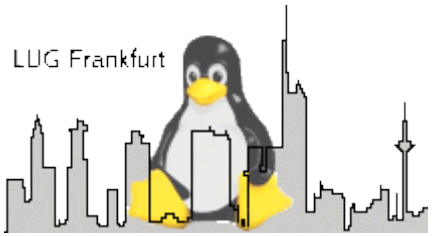
Einfache Shell Skripte (8a)

```
~$ chmod u+x ./eingabe2.sh
~$ . ./eingabe2.sh
Bitte eine Zeile eingeben: Hallo lieber Onkel Fritz.
Hallo
lieber
~$
```



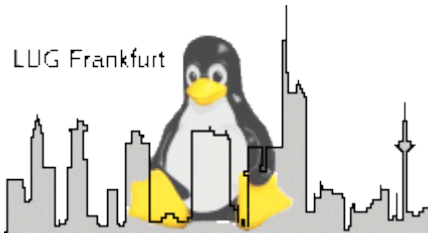
Einfache Shell Skripte (9)

```
~$ cat << EOF > mathe1.sh
#! /bin/bash
  echo $(($1 $2 $3))
EOF
~$ chmod u+x ./mathe1.sh
~$ . ./mathe1.sh  17 * 4
68
~$
```



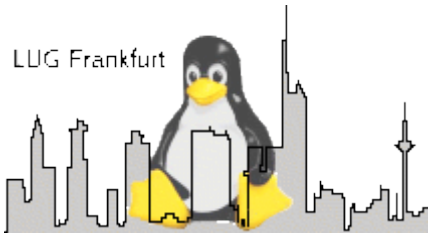
Einfache Shell Skripte (10)

```
~$ cat << EOF > mathe2.sh
#! /bin/bash
A=1
B=2
for i in 1 2 3 4 5 6
do
  echo -n "$A * $B = "
  A=$(( $A * $B ))
  echo -n „$A; “
done
EOF
~$ chmod u+x ./mathe2.sh
~$ . ./mathe2.sh
1 * 2 = 2; ... 32 * 2 = 64;
```



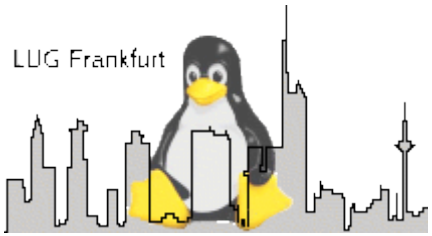
Einfache Shell Skripte (11)

```
~$ cat << EOF > mathe3.sh
#! /bin/bash
START=270
POS=${POS:=+60}
# Var. POS mit 60 vorbelegen, falls leer
CURRCOUNT=1
START=`expr $START - $CURRCOUNT '*' 37`
# Punkt vor Strich wird beachtet!
CURRCOUNT=`expr $CURRCOUNT '+' 1`
echo $START
EOF
~$ chmod u+x ./mathe3.sh
~$ ./mathe3.sh
233
~$
```



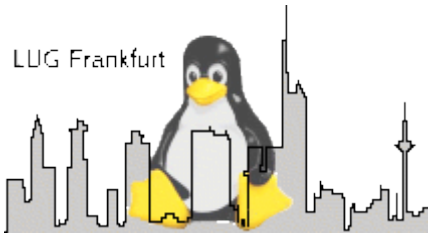
Einfache Shell Skripte (12)

```
~$ cat << EOF > mathe4.sh
#! /bin/bash
# Fließkommaberechnungen mit bc
WERT1=`echo “2.2 * 3.3“ | bc`
WERT2=`echo “sqrt(19.77)“ | bc`
# oder auf 10 Stellen genau...
WERT3=`echo “scale=10; sqrt(19.77)“ | bc`
echo $WERT1 $WERT2 $WERT3
EOF
~$ chmod u+x ./mathe4.sh
~$ . ./mathe4.sh
7.26 4.44 4.4463468150
~$
```



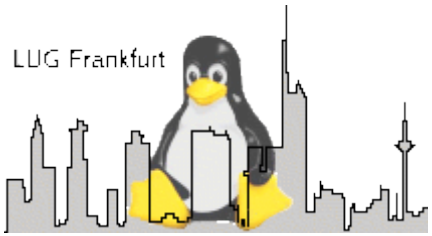
Aufbau des Vortrags

- Verschiedene Shells
- Besonderheiten der BASH
- Besonderheiten der Shell-Programmierung
- Einfache (Batch) Skripte
- Elemente von Programmiersprachen
- Einfache Shell-Skripte
- Komplexere Shell-Skripte
- Fragen und Diskussion



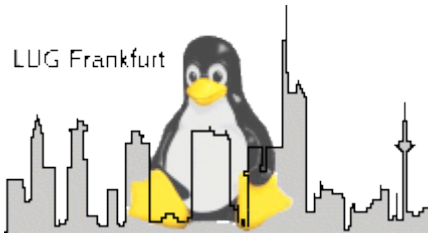
Komplexere Shell Skripte (1)

```
~$ cat << EOF > Skript4.sh
#! /bin/bash
# Befehlssubstitution
if [ $# -lt 1 ] ; then
    echo "Aufruf $0 (Telefon)nummer"
    exit
fi
echo $( $( grep $1 test | cut -f 2 - ) ).
EOF
~$ chmod u+x Skript4.sh
```



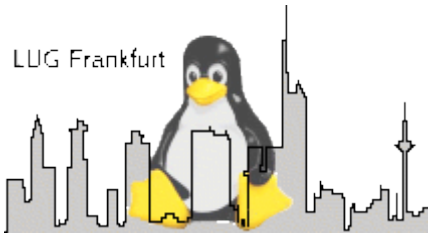
Komplexere Shell Skripte (1a)

```
~$ cat test
12345   echo "Hallo Hans"
33333   ls -l
54321   echo "Schön heute oder?"
~$ ./Skript4.sh 12345
Hallo Hans
~$
```



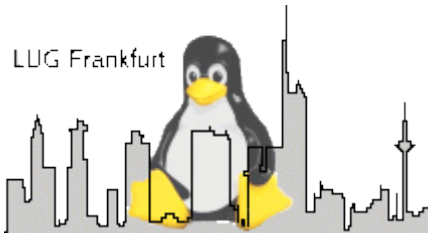
Komplexere Shell Skripte (2)

```
~$ cat << EOF > mytest.sh
#! /bin/bash
# Funktionen
function test ()
{
  echo $1
}
test $1
EOF
~$ chmod u+x mytest.sh
~$ ./mytest.sh "Hallo World"
Hallo World
~$
```



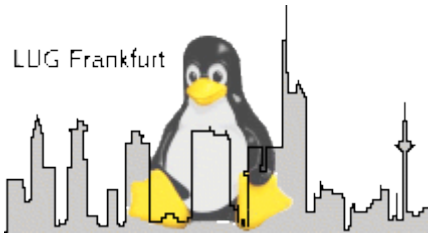
Aufbau des Vortrags

- Verschiedene Shells
- Besonderheiten der BASH
- Besonderheiten der Shell-Programmierung
- Einfache (Batch) Skripte
- Elemente von Programmiersprachen
- Einfache Shell-Skripte
- Komplexere Shell-Skripte
- Fragen und Diskussion



Literatur-Links

- Linux-Unix-Shells
  - Helmut Herold, „Bourne-Shell, Korn-Shell, C-Shell, bash,tcsh“ ISBN: 978-3-8273-1511-3
- Bash Guide for Beginners
  - von Machtelt Garrels
- Advanced Bash Scripting Guide
  - Mendel Cooper „An in-depth exploration of the art of shell scripting“
- Einführung in die bash-Shell
  - von Cameron Newham; ISBN: 978-3-89721-424-8



Ziel des Vortrags

- Sie sollten nun
  - die beiden Arten der Shell-Programmierung kennen
  - Möglichkeiten zum Einsetzen von Shell-Programmen im Alltag erkennen
  - Eigene Skripte erstellen können

Danke für Eure Aufmerksamkeit