unix / linux

```
#!/bin/sh -e
### BEGIN INIT INFO
# Provides:          ifupdown
# Required-Start:    ifupdown-clean
# Required-Stop:     $local_fs
# Default-Start:     S
# Default-Stop:      0 6
# Short-Description: Prepare the system for taking
### END INIT INFO

[ -x /sbin/ifup ] || exit 0
[ -x /sbin/ifdown ] || exit 0

. /lib/lsb/init-functions

MYNAME="${0##*/}"
report() { echo "${MYNAME}: $*" ; }
report_err() { log_failure_msg "$*" ; }
```

Praktische Shellskripte

Ziele

- Einige Beispiele für Shellskripte analysieren und verstehen
- Grundlegende Techniken der praktischen Shellprogrammierung kennen und anwenden können

Agenda

- **Benutzerdatenbank**
- Dateioperationen
- Systemadministration

1. Aufgabe

- Zähle zu einem Gruppennamen alle Benutzer auf, welche diese Gruppe als primäre Gruppe führen

Ressourcen

- `/etc/group`
  - GID für einen Namen
- `/etc/passwd`
  - Die primäre GID für den Benutzer

Erste Version

```
#!/bin/bash
# pgroup

# Hole die GID aus /etc/group
gid=$(grep "^$1:" /etc/group | cut -d: -f3)

# Suche nach den Benutzern mit der betreffenden
# GID in /etc/passwd
grep "^[^:]*:[^:]*:[^:]*:$gid:" /etc/passwd | \
 cut -d: -f1
```

Es fehlt ein Check, ob die Eingabe wirklich eine gültige Gruppe ist ...

1. Check: Syntaxüberprüfung

1. Check: gibt es einen Parameter?

```
if [ $# -ne 1 ]
then
    echo >&2 "usage: $0 GROUP"
    exit 1
fi

...
```

2.Check

2. Check: Ist die Gruppe wirklich eine Gruppe?
Warum wird mit [-z "$gid"] getestet, und nicht der Exitcode überprüft?

```
gid=$(grep "^$1:" /etc/group | cut -d: -f3)
if [ -z "$gid" ]
then
    echo >&2 "$0: group $1 does not exist"
    exit 1
fi
```

groups

Versuchen sie den Befehl `groups(1)` mit einem Shellskript zu implementieren.

groups(1) Resourcen

- `/etc/group`
  - GID für einen Namen
  - `/etc/passwd`
  - Die primäre GID für den Benutzer:

groups(1) Resourcen

- Primäre Gruppe

wie aus dem letzten Skript:

```
gid=$(grep "^$1:" /etc/passwd | cut -d: -f4)
grep "^[^:]*:[^:]*:[^:]*:$gid:" /etc/passwd | \
 cut -d: -f1
```

- Weiter Gruppen

```
grep "^[^:]*:[^:]*:[^:]*:.*\<$1\>" /etc/group |\
cut -d: -f1
```

Agenda

- Benutzerdatenbank
- **Dateioperationen**
- Systemadministration

2. Aufgabe

- ändere bei einer Gruppe von Dateien das Suffix

  Bsp.:

  bla.txt → bla.bac
  blo.txt → blo.bac

Ressourcen

- `mv`
  - Kann Dateien umbenennen
- `${variable%<muster>}`
  - Entfernt `<muster>` vom Ende von `variable`

Skript

```
#!/bin/bash
# chext

suffix="$1"
shift  # shift $1=$2; $2=$3; $n=$(n+1)

for f
do
    mv "$f" "${f%.*}.${suffix}"
done
```

Skript

```
# Besitzt der Dateiname keinen Punkt, der Pfade einen
# Punkt, so wird die Datei falsch umbenannt.
# dirname extrahiert den Pfad
# basename extrahiert den Dateinamen

for f
do
  d=$(dirname "$f");
  b=$(basename "$f");
  mv "$f" "$d/${b%.*}.${suffix}";
done
```

Check

```
# es müssen mindestens 1 Attribut und 1 Paramter an
# gegeben sein:

if [ $# -lt 2 ]; then
  echo >&2 "usage: $0 SUFFIX NAME ..."
  exit 1
fi
```

Agenda

- Benutzerdatenbank
- Dateioperationen
- **Systemadministration**

3. Aufgabe

Zeige die prozentuale Auslastung der Dateisystem als "Balkendiagramm" an

```
$ gdf
Mounted  Use%
/         91% ##########################################---
/dev/shm   0% ----------------------------------------------
/home     78% ###################################----------
/cdrom   100% ##############################################
```

Wie viele Shell-Zeilen sind dazu notwendig?

Ressourcen

- `df`
  - Liefert den aktuelle Auslastung der Festplatte
- `${name:p:n}`
  - Schneidet aus name ab p n Zeichen heraus

Skript - Vorversion

```
#!/bin/bash
# gdf

df | tr -s ' ' '\t' | cut -f5,6 | while read pct fs
do
  printf "%-12s %4s " $fs $pct
  # graphischer Teil
done
```

Grafische Ausgabe

```
hash="##############################################"
dash="----------------------------------------------"
if [ "$pct" != "Use%" ]; then
  usedc=$((${#hash}*${pct%\%}/100))
  echo "${hash:0:$usedc}${dash:0:${#hash}-$usedc}"
else
  echo ""
fi
```

all together ...

```
#!/bin/bash
# gdf

hash="##########################################"
dash="------------------------------------------"

df "$@" | tr -s ' ' '\t' | cut -f5,6 | while read pct fs
do
   printf "%-12s %4s " $fs $pct
   if [ "$pct" != "Use%" ]; then
      usedc=$((${#hash}*${pct%\%}/100))
      echo "${hash:0:$usedc}${dash:0:${#hash}-$usedc}"
   else
      echo ''
   fi
done
```

Zusammenfassung

- `grep` und `cut` sind nützlich, um bestimmte Zeilen und Spalten aus Dateien zu extrahieren.
- Sie sollten Fehlersituationen sorgfältig abfangen – sowohl beim Aufruf Ihres Skripts als auch während des Ablaufs.
- Ein- und Ausgabeumleitung für Folgen von Kommandos ist über explizite Subshells möglich.
- Die Kommandos `dirname` und `basename` erlauben Dateinamen-Manipulationen.
- Dateien mit Zuweisungen an Shellvariable können als bequeme "Konfigurationsdateien" für Shellskripte fungieren.
- Das `printf`-Kommando erlaubt die formatierte Ausgabe von textuellen oder numerischen Daten.

Fragen?