Some Bashing II - Mit der Kommandozeile Abläufe automatisieren

Linux-Infotag Augsburg

16. April 2016

Andreas Steil
Linux Consultant & Trainer
B1 Systems GmbH
steil@b1-systems.de

Some Bashing II: Mit der Kommandozeile Abläufe automatisieren

(Workshop-Unterlagen)

Agenda (Beispiele)

- Paar Grundlagen:
  - Bash
  - Shellscripts
    - Aufbau
    - Anwendung
    - Automatisierung
- Beispiele:
  1. Bilder automatisch umwandeln/verkleinern
  2. PDFs (z.B. aus Scans) zusammenführen
  3. Umlaute in HTML-Code konvertieren
  4. Konfigurationsdateien bereinigen
  5. Dateien automatisch umbenennen
  6. Musik abspielen mit der Kommandozeile mit eigenen Wiedergabelisten
  7. Id-Tags von MP3s auslesen und damit automatisch Musikdateien benennen
  8. ...

Beispiel 1: Bilder automatisch umwandeln/verkleinern

Bilder verkleinern: Ein (un)beholfener Weg

- „Es ist eine Krux mit den PDF-Dateien: Wer sie selbst erstellt – etwa beim Scannen – findet schnell heraus, dass die Dateigröße enorm ist. Wer Online-Bewerbungsmappen oder andere Dokumente per E-Mail verschicken will, stößt dabei schnell an die Grenzen gängiger Mailanbieter, zumal große E-Mails in vielen Unternehmen geblockt werden. Mit dem Online-Service SmallPDF gehört dieses Problem der Vergangenheit an: Das Online-Tool kann PDF-Dateien ganz eingach komprimieren, egal ob unter Windows, Mac OS X oder Linux. Alles, was Ihr dazu braucht, ist ein Browser – und ein wenig Upload-Bandbreite.“ Quelle: `http://www.tutonaut.de/tipp-pdf-dateien-schnell-und-einfach-auf-allen-systemen-verkleinern.html`
- Problem:
  „Natürlich gibt es bei der Nutzung von Onlinediensten dieser Art ein gewisses Risiko, dass die Daten von Dritten eingesehen werden. Der Dienst schrumpft die PDF-Dateien und verspricht, diese anschließend – im Stundentakt – vom Server zu löschen. ...“
- Lösung: Kommandozeilenprogramm `convert` (Paket „ImageMagick“)
  => kein Risiko mit der Bash ...

Bilder konvertieren mit convert (Beispiele)

- convert ⇒ Paket „ImageMagick“ (http://www.imagemagick.org/script/convert.php)

Bild in anderes Format konvertieren (im Beispiel ins PDF-Format):

```
# convert scan.jpg scan.pdf
```

Mini-Ansicht erstellen (im Beispiel 120x120 Pixel):

```
# convert -size 120x120 GroßesBild.tiff -resize 120x120 thumbnail.png
```

Bild verkleinern (im Beispiel auf 50%):

```
# convert bild.jpg -resize 50% bild.png
```

Bildgröße verändern (im Beispiel auf 768x1024 Pixel):

```
# convert -resize 768x1024 scan.jpg
```

Umwandeln eines kompletten Verzeichnisses

Umwandeln eines kompletten Verzeichnisses:

```
for BILD in *.jpg; \
  do    \
    convert -resize 800x600 $BILD ./kleiner/$BILD; \
  done
```

- `*.jpg` gibt das Format der Bilder an (alle gängigen Bildformate möglich)
- `-resize 800x600` gibt die neue Größe der Bilder an
- neue Bilder werden in (vorhandenem) Unterordner `./kleiner` gespeichert

(Der \ kennzeichnet einen Zeilenumbruch – wegen der übersichtlicheren Darstellung.)

Beispiel 2: PDFs (z.B. aus Scans) zusammenführen

PDFs zusammenführen mit pdftk

- Dokumentationen
- Bewerbungen
- Hausarbeiten
- Scans
- ...

Erst alles ins gleiche Format überführen (im Beispiel PDF):

```
# convert Zeugnis_1.jpg Zeugnis_1.pdf
```

...dann zusammenführen:

```
# pdftk Anschreiben.pdf Lebenslauf.pdf Zeugnis_1.pdf \
cat output Bewerbung.pdf
```

...oder:

```
# pdftk Hausarbeit.pdf Eigenständigkeitserklärung.pdf \
  cat output Hausarbeit_John_Doe.pdf
```

Skripte (sehr einfache Einführung)

- Skript = Befehle in Textdatei zum wiederholten Aufrufen
- Skript ausführbar machen: `chmod +x <skriptname>`
- Skript aufrufen: `<skriptname>` (`/bin/bash <skriptname>`)
- bei Bash-Skripten erste Zeile `#! /bin/bash` (Shebang-Anweisung) ⇒ ausführender Kommandointerpreter

Einfaches, aber freundliches Skript erstellen (Beispiel):

```
$ echo Hallo Du da\! > hallo.sh
```

Skript ausführbar machen:

```
$ chmod +x hallo.sh
```

Skript aufrufen/ausführen:

```
$ hallo.sh
Hallo Du da!
```

Skript zum Zusammenführen von PDFs mit `for`-Schleife

Skript zum PDFs zusammenführen:

```
#! /bin/bash

convert ~/LIT_2016/book/cover.jpg ~/LIT_2016/gesamt/gesamt.pdf

for i in $(ls ~/LIT_2016/book)
do
   echo $i wird verarbeitet ...
   convert ~/LIT_2016/book/$i ~/LIT_2016/gesamt/$i.pdf
   pdftk ~/LIT_2016/gesamt/gesamt.pdf ~/LIT_2016/gesamt/$i.pdf \
         cat output ~/LIT_2016/gesamt/gesamt_1.pdf
   mv ~/LIT_2016/gesamt/gesamt_1.pdf ~/LIT_2016/gesamt/gesamt.pdf
done

echo Ich habe fertig!
okular ~/LIT_2016/gesamt/gesamt.pdf &
```

Bequemer & übersichtlicher: Suchpfad in eigenes Skriptverzeichnis

- In der Umgebungsvariablen `PATH` sind Verzeichnisse als (Such-)Pfade für Befehle (auch Skripte) definiert.
- kann neu definiert und dadurch erweitert/geändert werden (Trennzeichen: `:`)
- Pfadangabe zum Skript kann so beim Aufruf erspart werden.
- dauerhaftes Ändern der Umgebungsvariable `PATH` z. B. im Anmeldeskript des Users `~/.bashrc`

Umgebungsvariable `PATH` abfragen:

```
$ echo $PATH
/home/<user>/bin:/usr/local/bin:/usr/bin:/bin
```

Umgebungsvariable `PATH` um Verzeichnis erweitern (temporär):

```
$ PATH=$PATH:/home/b1/LIT_2016/scripte
```

Beispiel 3: Umlaute in HTML-Code konvertieren

Vorspann: etwas sed

- `sed` = Stream Editor
- Syntax: z. B. `sed 's/regexp/replacement/' ...`
- hilfreiche Befehle/Optionen:
  - `s` ersetzen (substitute)
  - `g` jedes (nicht nur erstes) Vorkommen (global)
  - `-e` Verknüpfung mehrerer Editierbefehle (z. B. `sed -e '<befehl1>' -e '<befehl2>'`; ⇒ Besipiel 5)
  - `-f` Anwendung auf angegebene Datei statt Stream (file)

Anwendung (Beispiel):

```
# echo abc | sed 's/abc/yxz/'
yxz
```

sed – Editierbefehle

Einige Editierbefehle von sed

Befehl	Funktion
a	Einfügen nach der aktuellen Zeile
i	Einfügen vor der aktuellen Zeile
d	Löschen
p	Ausgeben
c	Zeilen ersetzen
s	Suchen und Ersetzen
y	Zeichen durch andere Zeichen ersetzen

Suchen und Ersetzen mit sed

- eine der wichtigsten Funktionen von sed: Suchen & Ersetzen
- Suchbegriff meist regulärer Ausdruck
- nur erstes Vorkommen pro Zeile wird ersetzt, es sei denn, Sie verwenden Parameter g (*global*)

Erstes Vorkommen pro Zeile wird ersetzt

```
$ sed 's/Latex/LaTeX/' datei
```

Alle Vorkommen werden ersetzt

```
$ sed 's/Latex/LaTeX/g' datei
```

Anwendung von sed – Beispiele

Beispiel: Zeile 12 bis Ende der Datei löschen

```
$ sed '12,$d' datei
```

Beispiel: Zeile 5 durch „lalala“ ersetzen

```
$ sed '5c lalala' datei
```

Bestimmte Buchstaben ersetzen

```
$ echo "P1 Sysdems" | sed -e 'y/Pd/Bt/'
```

(Konfigurations-)Dateien suchen und Kommentare entfernen

```
# find /<pfad>/ -type f -iname "*.conf" -exec sed -i '/^#/d' '{}' \;
```

Umlaute in HTML-Code konvertieren mit sed

Anwendung von sed (ein Buchstabe):

```
# echo Örömüz Ürgümül! > umlaute.html
# sed 's/ü/\&uuml;/g' umlaute.html
Öröm&uuml;z Ürg&uuml;m&uuml;l!
```

Anwendung von sed (mehrere Buchstaben):

```
# sed -e 's/Ö/\&Ouuml;/g' -e 's/ö/\&ouml;/g' \
      -e 's/Ü/\&Uuml;/g' -e 's/ü/\&uuml;/g' \
      [...] \
      umlaute.txt
&Ouuml;r&ouml;m&uuml;z &Uuml;rg&uuml;m&uuml;l!
```

...einfacher per Skript, gespeist aus Sed-File:

Sed-File (z.B. ersetzungen):

```
s/ä/\&auml;/g
s/ö/\&ouml;/g
...
```

Skript (z.B. ersetze) für Sed-File (mit Sicherungskopie):

```
for i in $*
  do
     cp $i $i.bak
     sed -f ./ersetzungen < $i.bak > $i
  done
echo "Ich habe fertig !"
```

Anwendung:

```
# ersetze umlaute.html
Ich habe fertig !
```

Das Ganze auf einem anderen Rechner . . . (via ssh):

Skript (z.B. umls_auf):

```
#! /bin/bash
read -p "Host: " HOST
read -p "User: " USER
read -s -p "Passwort:" PW
read -p "Datei: " DATEI
sshpass -p $PW ssh $USER@$HOST \
        -t "sh /pfad/zum/skript/ersetze $DATEI"
```

Skript ausführen:

```
$ umls_auf
Host: 8.8.8.8
User: webling
Passwort:
Connection to 8.8.8.8 closed.
```

Beispiel 4: Konfigurationsdateien bereinigen

Zeichenketten ermitteln mit grep

- `grep` = *global regular expression print*
- Syntax: z. B. `grep [OPTIONS] PATTERN [FILE...]`
- hilfreiche Befehle/Optionen:
  - `-R` auch Unterverzeichnisse durchsuchen (recursive)
  - `-i` Groß- und Kleinbuchstaben nicht unterscheiden (ignore)
  - `-v` alle Zeilen ausgeben, in denen der reguläre Ausdruck nicht erfüllt ist

Anwendung (Beispiel): 'sepp mit grep':

```
# grep -iR sepp
namen.txt:sepp
```

⇒ in der Datei `namen.txt` gibt's einen Treffer für `sepp`

Kommentare entfernen mit grep

Kommentarzeilen entfernen mit grep (Beispiel; Ausgabe auf Bildschirm):

```
# grep ^[^#] /etc/xinetd.conf
defaults
{
log_type = SYSLOG daemon info
log_on_failure = HOST ATTEMPT
log_on_success = HOST EXIT DURATION
[...]
groups = yes
umask = 002
}
includedir /etc/xinetd.d
```

⇒ Standardausgabe

(^ = am Anfang; regulärer Ausdruck in [...],

hier: [^#] = nicht Kommentarzeichen;

⇒ alle Zeilen, bei denen am Anfang *nicht* ein Kommantarzeichen steht)

Kommentare entfernen mit grep (in Datei)

. . . als interaktives Skript (Ausgabe in Datei):

```
#! /bin/bash
echo "Dateinamen eingeben:"
read DATEI
cp $DATEI $DATEI.orig
grep ^[^#] $DATEI.orig > $DATEI
echo Kommentarzeilen in Datei $DATEI wurden entfernt, \
     das Original als $DATEI.orig gesichert.
```

⇒ Ausgabe in Datei

(`read DATEI` erwartet Benutzereingabe, die in der Variablen `$DATEI` gespeichert wird.)

Dateien suchen mit `find`

z. B. alle Dateien mit der Endung `.conf` in `/etc` suchen:

```
# find /etc -name *.conf
```

. . . beides zusammen als Skript: (für ein bestimmtes Verzeichnis)

Skript für das Verzeichnis /etc:

```
#! /bin/bash

for DATEI in ‘find /etc -name "*.conf"‘
do
  echo Datei $DATEI gefunden ...
  cp $DATEI $DATEI.orig
  grep ^[^#] $DATEI.orig > $DATEI
  echo Kommentarzeilen in Datei $DATEI wurden entfernt, \
       das Original als $DATEI.orig gesichert.
done
```

Wichtiger Hinweis: Nur als Beispiel gedacht!
(Nicht zweimal ausführen, sonst werden die Sicherungen (*.conf.orig) mit der kommentarlosen Version überschrieben.)

. . . beides zusammen als Skript: (Version 2: interaktiv mit Benutzereingabe)

Interaktives Skript (Benutzereingabe mit `read`):

```
#! /bin/bash
echo "Bitte Verzeichnis angeben: "; read VERZEICHNIS
for DATEI in $(find $VERZEICHNIS -name "*.conf")
do
  echo Datei $DATEI gefunden ...
  cp $DATEI $DATEI.orig
  grep ^[^#] $DATEI.orig > $DATEI
  echo Kommentarzeilen in Datei $DATEI wurden entfernt, \
       das Original als $DATEI.orig gesichert.
done
```

Wichtiger Hinweis: Nur als Beispiel gedacht!
(Nicht zweimal ausführen, sonst werden die Sicherungen (`*.conf.orig`) mit der kommentarlosen Version überschrieben.)

. . . beides zusammen als Skript: (Version 3: mit Parameterübergabe)

Skript mit Parameterübergabe ($1 = 1.Parameter):

```
#! /bin/bash

for DATEI in ‘find $1 -name "*.conf"‘
do
  echo Datei $DATEI in Verzeichnis $1 gefunden ...
  cp $DATEI $DATEI.orig
  grep ^[^#] $DATEI.orig > $DATEI
  echo Kommentarzeilen in Datei $DATEI wurden entfernt, \
       das Original als $DATEI.orig gesichert.
done
```

Wichtiger Hinweis: Nur als Beispiel gedacht!
(Nicht zweimal ausführen, sonst werden die Sicherungen (`*.conf.orig`) mit der kommentarlosen Version überschrieben.)

Beispiel 5: Dateien automatisch umbenennen

Problem: Leerzeichen in Datei- und Verzeichnisnamen

Problem: Leerzeichen in Datei- und Verzeichnisnamen

- Leerzeichen ⇒ Metazeichen mit Sonderbedeutung
- werden oft anders interpretiert (z. B. als Trennzeichen in der Bash)
- müssen maskiert werden (z. B. mit Backslash in der Bash)
- oft unübersichtlich
- ...

Beispiel: Datei mit Leerzeichen (maskiert mit Backslash) anlegen:

```
# touch la\ la\ \ la
# ls
la la  la
```

Leerzeichen in Texten durch Unterstrich ersetzen

Leerzeichen durch Unterstrich ersetzen mit `tr`:

```
# echo "la la    laa" | tr -s ' ' '_'
la_la_laa
```

Option `-s` (`--squeeze-repeats`)
⇒ Wiederholungen von Zeichen werden als ein Zeichen gewertet

Leerzeichen durch Unterstrich ersetzen mit `sed`:

```
# echo "la la    la" | sed -e 's/  */_/g'
la_la_la
```

- `*` ⇒ Leerzeichen, beliebig oft
- `g` (`global`) ⇒ für alle Vorkommen

⇒ nächster Schritt: für Datei- und Verzeichnisnamen

Lösung 1: Parameterexpansion der Bash

Leerzeichen durch Unterstrich ersetzen mit Parameterexpansion:

```
# for DATEI in ./*; do
    mv "$DATEI" "${DATEI// /_}"
  done
```

- `${DATEI }`
  ⇒ weist die Bash an, den Inhalt der Variable `$DATEI` zu expandieren
- der / ist der Operator für Suchen und Ersetzen
  ⇒ `${VarName/A/B}` ersetzt das *erste* Vorkommen von A im Inhalt von `$VarName` durch B
- `${VarName//A/B}` ersetzt *alle* As durch Bs

Lösung 2: Skript mit sed

Leerzeichen durch Unterstrich ersetzen mit sed (als Skript):

```
DIR=$1
for i in $DIR/*
do
        ALTER_NAME=$i
        NEUER_NAME=‘echo $i | sed ’s/ /_/g’‘
        mv $ALTER_NAME $NEUER_NAME
        echo $ALTER_NAME wurde in $NEUER_NAME umbenannt.
done
```

$1 ⇒ 1.Parameter wird übernommen (/pfad/zum/verzeichnis)

Aufruf:

```
# /pfad/zum/script /pfad/zum/verzeichnis
verzeichnis/la la la wurde in verzeichnis/la_la_la umbenannt.
```

. . . noch schöner machen . . .

Lösung 2: Skript ausbauen

. . . Skript ausbauen:

- für Dateien und Verzeichnisse
- auch für mehrere Parameter
- rekursive Wirksamkeit
- Prüfen auf bereits vorhandene Dateien und Verzeichnisse
- Hilfe-Funktion
- Formales: Shebang-Anweisung, Beschreibung, Autor, . . .
- . . .

Leerzeichen durch _ ersetzen: Ein perfektes Skript-Beispiel 1/2

. . . als vorbildliches Skript von Anke Börnig (Teil 1/2):

```
#!/bin/bash

# Dieses Skript sucht in den angegebenen Verzeichnissen zuerst nach Unter-Verzeichnissen,
# die ein Leerzeichen im Namen haben, und ersetzt die Leerzeichen durch "_".
# Anschliessend passiert das gleiche fuer Dateinamen.

# Autor: Anke Boernig, anke@boernig.de # Datum: 29.05.02

SCRIPTNAME=‘basename $0‘
# Hilfefunktion fuer Hilfetext
hilfe () {
  cat << EOF
Benutzung: $SCRIPTNAME Verzeichnis1 [Verzeichnis2 ...]
Das Skript $SCRIPTNAME wandelt alle Leerzeichen innerhalb von
Verzeichnis- und Dateinamen in den angegebenen Verzeichnissen in "_" um.
EOF
}

if [ "$#" -lt 1 ]
then
  hilfe
  exit 0
fi
```

Leerzeichen durch _ ersetzen: Ein perfektes Skript-Beispiel 2/2

. . . als vorbildliches Skript von Anke Börnig (Teil 2/2):

```
for VERZ in $@
do
  for TYP in d f
  do
    while ORIG=$(find $VERZ -type $TYP -name "* *" | head -n1 | grep '.* .*')
    do
      NEU=$(echo $ORIG | sed 's/ /_/g')
      while test -e $NEU
      do
        PFAD=${NEU%/*}
        echo "Ein(e) Verzeichnis/Datei $NEU existiert schon."
        echo "Bitte einen anderen Namen eingeben:"
        read NAME
        NEU=$PFAD/$NAME
      done
      echo "$ORIG wird umbenannt nach $NEU."
      mv "$ORIG" "$NEU"
    done
  done
done
```

Problem: sehr viele Dateien umbenennen

Problem: sehr viele MP3s umbenennen, die Namen haben wie `Pippi_Langstrumpf_in_Taka_Tuka_Land-01.mp3` bis `Pippi_Langstrumpf_in_Taka_Tuka_Land-23.mp3`

„Da die blöden MP3-Player meiner Kinder aber zu doof sind, so lange Dateinamen auszuwerten, und dadurch die Reihenfolge immer durcheinander ist, müsste ich ein paar Hundert Dateien umbenennen, so dass die Zahlen vorne stehen, also z. B.
`Pippi_Langstrumpf_in_Taka_Tuka_Land-01.mp3 ->`
`01-Pippi_Langstrumpf_in_Taka_Tuka_Land.mp3` bis

`Pippi_Langstrumpf_in_Taka_Tuka_Land-23.mp3 ->`
`23-Pippi_Langstrumpf_in_Taka_Tuka_Land.mp3`“

Lösung: sed

Dateien automatisch umbenennen (for-Schleife mit mv und sed):

```
# for i in $(ls *.mp3); \
do mv $i $(echo $i | \
sed -e 's/\([a-z,A-Z,_]\{0,\}\)-\([0-9]\{0,\}\)\.mp3/\2-\1\.mp3/'); \
done
```

... etwas aufgedröselt (in Skript-Form):

```
#! /bin/bash
# for-Schleife: für alle MP3s
# (ls-Befehl könnte auch mit Verzeichnisangabe erfolgen)
for i in $(ls *.mp3)
# innerhalb der for-Schleife: Umbenennen mit 'mv',
# wobei $i = alter Name, $(echo ...) = neuer Name
do mv $i $(echo $i |  \
sed -e 's/\([a-z,A-Z,_]\{0,\}\)-\([0-9]\{0,\}\)\.mp3/\2-\1\.mp3/')
done
```

sed-Ausdruck (etwas entwirrt)

Der sed-Ausdruck:

```
sed -e 's/\([a-z,A-Z,_]\{0,\}\)-\([0-9]\{0,\}\)\.mp3/\2-\1\.mp3/')
```

Ersetze Dateinamen (s = substitute):

```
[a-z,A-Z,_]\{0,\}\)-\([0-9]\{0,\}\)\.mp3
```

⇒ erst alles bis zum Bindestrich (= alle Buchstaben einschl. Unterstrich) als 1.Teil abtrennen (⇒ \1); nachfolgende Nummerierung wird 2.Teil (⇒ \2)

. . . mit:

```
\2-\1\.mp3
```

⇒ der zweite Teil (\2) von oben (= Nummer) wird – durch Bindestrich getrennt – nach vorne (= vor den ersten Teil \1) gestellt

Zum Nachmachen

100 Fake-Dateien zum Testen anlegen:

```
# touch Pippi_Langstrumpf_in_Taka_Tuka_Land-{01..99}.mp3
```

...

Beispiel 6: Musik abspielen mit der Kommandozeile und eigene Wiedergabelisten

Exkurs: Freie Musik im Internet

Freie Musik im Internet (kleine Auswahl):

- `http://www.ccc-r.de/Boycott_musicindustry`
- `https://soundcloud.com/`
- `https://www.medienpaedagogik-praxis.de/kostenlose-medien/freie-musik/`
- ...

The WIRED CD: Rip. Sample. Mash. Share.

- „The WIRED CD: Rip. Sample. Mash. Share.“ => freie Musik !
- URL: `http://creativecommons.org/wired/`
- „These musicians are saying that true creativity needs to be open, fluid, and alive. When it comes to copyright, they are pro-choice.“

„The WIRED CD“ runterladen mit `wget`:

```
# wget -r -e robots=off \
  http://mirrors.creativecommons.org/ccmixter/contrib/Wired/
[...]
BEENDET --2016-04-14 12:02:00--
Verstrichene Zeit: 3m 47s
Geholt: 100 Dateien, 622M in 3m 38s (2,85 MB/s)
```

Musik abspielen mit der Kommandozeile

Tools (Auswahl):

- `cmus`
- `cvlc`
- `mpc`
- `mpd`
- `mpg123`
- `mocp`
- `ncmpcpp` (mpd Frontend)
- `ncmpcppcppcpp`
- ...

Musik abspielen mit mpg123

Musik abspielen (* ⇒ alle Dateien im aktuellen Verzeichnis; & ⇒ im Hintergrund):

```
# mpg123 * &
```

Musik abspielen mit mpg123 (Beispiel):

```
# mpg123 mirrors.creativecommons.org/ccmixter/contrib/Wired/* &
[1] 11267
b1@:~/LIT_2016/music> High Performance MPEG 1.0/2.0/2.5 Audio Player
version 1.22.4; written and copyright by Michael Hipp and others
free software (LGPL) without any warranty but with best wishes

Directory: mirrors.creativecommons.org/ccmixter/contrib/Wired/
Playing MPEG stream 1 of 50: Beastie Boys - Now Get Busy.mp3 ...
MPEG 1.0 layer III, VBR, 44100 Hz joint-stereo
Title:   Now Get Busy                    Artist: Beastie Boys
Comment:
Album:   The Wired CD: Rip. Sample. Mash. Share.
Year:    2004                            Genre:  Hip-Hop
```

Musik abspielen mit der Kommandozeile: Wiedergabelisten (Playlists) als einfache Textdateien

- einfache Textdateien sind universeller als proprietäre Playlisten
- Nach- / Weiterverarbeitung mit allen Textwerkzeugen möglich

Wiedergabeliste als einfache Text-Datei, z. B. `playlist-1`:

```
lied1.mp3
/pfad/zum/lied2.mp3
/musik/noch_ein_lied.mp3
[...]
```

Wiedergabeliste abspielen (z. B. mit mpg123)

Playlist mit mpg123-Optionen:

```
# mpg123 -@ playlist-1
```

- Option `-@`: ⇒ Dateinamen einlesen (hier aus der Datei `playlist`; auch URL möglich!)
- Option `-Z`: ⇒ zufällige Wiedergabe (Shuffle-Mode)

. . . oder auch als for-Schleife:

```
# for I in $(cat playlist-1); do mpg123 $I; done
```

Beispiel 7: Id-Tags von MP3s auslesen und damit automatisch Musikdateien benennen

Id-Tags von MP3s auslesen

Tools zum Auslesen von Id-Tags (kleine Auswahl):

- `eyeD3`
- `id3tool`
- `id3info`
- `mp3info`
- `mid3v2` (`mutagen`)
- `...`

Id-Tags von MP3s auslesen mit mid3v2

Id-Tags von MP3s auslesen mit mid3v2:

```
# mid3v2 -l ~/Musik/*.mp3
[...]
TALB=The Wired CD: Rip. Sample. Mash. Share.
TCOP=2004 David Byrne Licensed to the public under \
     http://creativecommons.org/licenses/sampling+/1.0/ \
     verify at http://ccmixter.org/file/Wired/61
TIT2=My Fair Lady
TPE1=David Byrne
TPUB=Creative Commons
TRCK=2
TYER=2004
```

Beispiel: Titel extrahieren mit grep und cut

```
# mid3v2 David\ Byrne\ -\ My\ Fair\ Lady.mp3 | grep TIT2 | \
  cut -d'=' -f 2
My Fair Lady
```

. . . usw. . . .

Skript: Mit Id-Tags automatisch Musikdateien benennen

1. gewünschte Tags auslesen (z. B. auch mit `awk`)
2. Tags zu neuem Namen zusammensetzen
3. Dateien umbenennen
4. ...

usw.
... oder ein fertiges Shell-Skript verwenden ...
(z. B. `abcde` – kann Id-Tags auch aus Internet-Datenbank beziehen)

CDs auslesen mit abcde

- Shell-Skript `abcde` („A Better CD Encoder“) = Frontend für andere Kommandozeilen-Tools
- `abcde` und nötige Systemkomponenten:
  - Grabber zum Auslesen der Musikstücke aus Audio-CDs (z. B. `cdparanoia` oder `cdda2wav`)
  - Encoder zum Generieren von MP3- oder Ogg-Vorbis-Dateien (`oggenc`, `vorbize`, `lame`, ...)
  - CDDB-Tool (z. B. `cd-discid`)
  - Programm zum Spiegeln von Daten aus dem WWW (z. B. `wget`)
  - optional: Programm zu Setzen von ID3-Tags (z. B. `id3v2`)

CDs auslesen mit abcde:

```
# abcde
Grabbing entire CD - tracks: 01 02 03 04 05 06 07 08 09 10 11 12 13
Retrieving 1 CDDB match...done.
[...]
```

Ein paar Links:

1. Link zum Vortrag „Some Bashing I“ (LIT 2015):

   `http://www.luga.de/Angebote/Vortraege/Some_Bashing_LIT_2015`
2. abcde - A Better CD Encoder:

   `https://abcde.einval.com`

   Shell-Skript, das auf der Kommandozeile CDs ausliest, die Tracks in MP3s oder Oggs konvertiert und auf Wunsch ID3-Tags hinzufügt
3. Bash und Shellscripts:
   - Shell programmieren mit Bash Scripts:
     `http://www.strassenprogrammierer.de/shell-programmieren-mit-bash-scripts-unter-linux_tipp_598.html`
   - BASH Programming - Introduction HOW-TO:

     `http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html`
   - Advanced Bash-Scripting Guide:

     `http://www.tldp.org/LDP/abs/html/`

Vielen Dank für Ihre Aufmerksamkeit!

Bei weiteren Fragen wenden Sie sich bitte an info@b1-systems.de
oder +49 (0)8457 - 931096