Kanäle

Datei `ausgabe.txt` mit Standard-Ausgabe des Befehls `ls -la` überschreiben:

```
ls -la > ausgabe.txt
cat ausgabe.txt
```

→ Länge der Datei `ausgabe.txt` beachten!

Standard-Ausgabe des Befehls `date` an Datei `ausgabe.txt` anhängen:

```
date >> ausgabe.txt
cat ausgabe.txt
```

Standard-Eingabe:

```
wc -l ausgabe.txt
wc -l < ausgabe.txt
wc -l /etc/shadow
wc -l < /etc/shadow
```

Datei `fehler.txt` mit Fehlerausgabe des Befehls `ls gibtsnicht` überschreiben:

```
ls gibtsnicht 2> fehler.txt
```

Fehlerausgabe des Befehls `ls gibtsnicht` an Datei `fehler.txt` anhängen:

```
ls gibtsnicht 2>> fehler.txt
```

```
ls . gibtsnicht 2> fehler.txt
ls . gibtsnicht > ausgabe.txt
ls . gibtsnicht >& beides.txt
ls . gibtsnicht > beides.txt 2>&1
```

Pipes

Beispiele:

```
dpkg -l | wc -l
dpkg -l | tee pakete.txt | less
```

Doku:

- Pipe (Informatik)
- detaillierte, technische Beschreibung, wie Pipes implementiert sind

Filterbefehle

cat, tac & split

Tabelle 8.2 im Skript

```
tar cz --directory /usr/share doc/ | split -b 10M - doc.tar.
cat doc.tar.a* | tar tz | tail
```

Benutzerliste rückwärts und mit Zeilennummern

```
tac /etc/passwd
cat -n /etc/passwd
```

head & tail

```
head -n 20 /etc/services
tail -n 20 /etc/services
```

Kapitel 8.3.2 im Skript lesen

Übungen 8.8 - 8.11 im Skript

od

```
echo 'Müller' | od -a
```

cut

```
cut -d: -f 7 /etc/passwd
```

sort

Tabelle 8.10 im Skript

Übungen 8.22, 8.23, 8.26 im Skript

uniq

```
find /usr/share/doc -printf '%y\n' | sort | uniq -c
```

grep

```
cat /etc/passwd | grep bash
```

[1)]

column

```
column -t /etc/fstab
ip -o link | column -t
ip -o address | column -t
column -s: -t /etc/passwd
```

Pipeline

Statistik aller eingegebenen Befehle

```
history | tr -s "[[:space:]]" | cut -d" " -f 3- | sort | uniq -c| sort -nr
```

Übung

Wer errät, wofür dieser Befehl gut ist? [2)]

```
seq 1 6 | shuf | head -n 1
```

Vorbereitung des Beispiels

Testdaten ins Heimatverzeichnis kopieren:

```
cd
cp -a /usr/share/doc .
```

Ausgabe eines Befehls in Datei umleiten

jpg-Bilder in `doc` suchen und Ergebnis in `bilder.txt` speichern:

```
find doc/ -xdev -name "*.jpg" > /tmp/bilder.txt
```

Das Selbe, aber Fehlermeldungen unterdrücken:

```
find doc/ -xdev -name "*.jpg" > /tmp/bilder.txt 2> /dev/null
```

Dateiinhalt mit Befehl weiterverarbeiten

Berechtigung der in der Datei `bilder.txt` aufgeführten Dateien mit `chmod` ändern:

```
xargs chmod g+w < /tmp/bilder.txt
```

`xargs` liest die (durch ein Zeilenende `\n` getrennten) Dateinamen (ggf. mit Pfad) aus der Standardeingabe und hängt diese als Liste an das Kommando (hier `chmod g+w`) an.

Ausgabe eines Befehls mit zweitem Befehl weiterverarbeiten

jpg-Bilder in `doc` suchen und Berechtigung mit `chmod` ändern:

```
find doc/ -xdev -name "*.jpg" 2> /dev/null | xargs chmod g-w
```

Verbesserte Version, die auch mit Leer- und Sonderzeichen in Dateinamen klar kommt:

```
mv doc 'Eigene Dateien'
find Eigene\ Dateien -xdev -name "*.jpg" -print0 2> /dev/null | xargs -0
chmod g+w
mv Eigene\ Dateien doc
```

Alle JPEG-Grafikdateien in das Verzeichnis Bilder kopieren:

```
find / -name "*jpg" -exec cp {} Bilder/ \; 2>/dev/null
```

Hier führt `find` für jede gefundene Datei das Kommando `cp` aus. Bei sehr vielen Dateien kann das sehr langsam werden.

```
find / -name "*jpg" -print0 2> /dev/null | xargs -0 cp -t Bilder
```

Hier schreibt `find` die gefundenen Namen durch ein Nullbyte `\0` getrennt in die Pipe, aus der `xargs` liest. Durch die Option `-0` erwartet `xargs`, dass die Dateinamen nicht durch ein Zeilenende (`\n`), sondern durch ein Nullbyte getrennt sind. `xargs` baut für `cp` eine lange Parameterliste aus

Dateinamen zusammen und übergibt sie. Falls mehr Namen ankommen als auf eine Kommandozeile passen, wird cp ggf. mehrfach aufgerufen, aber viel seltener als im vorigen Beispiel.

Weil cp üblicherweise das Ziel als letzten Parameter erwartet, muss hier die Option -t verwendet werden, um das Zielverzeichnis zu Beginn anzugeben.

Eine Variante ohne xargs und ohne das cp-Kommando für jede gefundene Datei aufzurufen:

```
find / -name "*jpg" -exec cp -t Bilder/ {} + 2>/dev/null
```

Das +-Zeichen braucht nicht geschützt zu werden. Es werden ganz viele Dateinamen am Ende des Kommandos anstelle des {} eingefügt.

Fehlerausgabe und Standardausgabe zusammenfassen

Lange Version, geht auch mit anderen Shells als bash:

```
find doc/ -xdev -name "*.jpg" > /tmp/bilder.txt 2>&1
```

Nur mit bash:

```
find doc/ -xdev -name "*.jpg" &> /tmp/bilder.txt
```

Fehlerausgabe und Standardausgabe mit zweitem Befehl weiterverarbeiten

```
find /var -xdev -type d -ls 2>&1 | less
```

Nur mit bash:

```
find /var -xdev -type d -ls |& less
```

Todo: Beispiel für tee

```
tee ausgabe1 ausgabe2 < /etc/passwd | wc -l
wc -l ausgabe1 ausgabe2
```

Sonderfall: stdout und stderr unterschiedlich weiterverarbeiten

```
((ls -l null eins |nl) 3>&1 1>&2 2>&3 | grep -v 'spezielle fehlermeldung' )
3>&1 1>&2 2>&3
```

Es werden dazu stdout und stderr zweimal reihum getauschst.

Links

- http://www.tldp.org/LDP/abs/html/io-redirection.html

1)

Moderne Alternativen:

- ripgrep
- ugrep

2)

https://massimo-nazaria.github.io/blog/2019/03/02/unix-philosophy-with-an-example.html