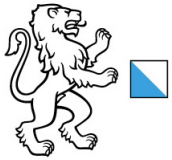

Bildungszentrum Uster
Berufsfachschule
Uster
Wirtschaft und Technik

Linux Shell - Lektion 2

Mario Bischof

Berufsfachschule Uster

mario.bischof@bzu.ch

November 9, 2015

Übersicht

Informationskanäle

Wenn sie mit der Shell arbeiten, gibt es unterschiedliche Informationskanäle, welche sie verwenden können.

- `stdin` - Standardeingabekanal (0)
  (zB. sie geben Zeichen über die Tastatur ein)
- `stdout` - Standardausgabekanal (1)
  (zB. ein Programm zeigt den Inhalt eines Verzeichnisses am Bildschirm an)
- `sterr` - Standardfehlerausgabekanal (2)
  (zB. ein Programm erzeugt einen Fehler und zeigt diesen am Bildschirm an)

Jeder der Kanäle kann über die jeweilige Nummer angesprochen werden (0,1,2)

Ausgabe umleiten

Die Ausgabe eines Befehls kann umgeleitet werden mit > oder >>
Beispiele:

- `ls -la > liste.txt`
- `./meinskript > outputofscript.txt`
- `cat outputofscript.txt >> list.txt`

>> hängt Inhalt an bestehende Datei an, > überschreibt den Inhalt komplett mit Neuem

Ausgabe umleiten

Die unterschiedlichen Kanäle können mit der Nummer spezifiziert werden:

- `./meinskript 2> errorsofscript.txt`
  (Leitet nur Fehlermeldungen in die Datei `errorsofscript.txt`)
- `./meinskript 1> outputofscript.txt`
  (Leitet den üblichen Output in die Datei `outputofscript.txt`)
- `./meinzweitesskript 2>> errorsofscript.txt`
  (Dasselbe geht auch im Anhängemodus)
- `./skript 1> output.txt 2> error.txt`
  (Unterschiedliche Umleitungen der Kanäle in einem Befehl)

Ausgabekanäle zusammenlegen / Ausgaben unterdrücken

Will man Standardausgabe und Standardfehlerausgabe über denselben Kanal ausgeben, kann man diese mit `2>&1` (Leitet `stderr` in `stdout`) koppeln:

```
./skript > output.txt 2>&1
```

Will man einen Ausgabekanal *ausschalten*, kann dieser nach `/dev/null` (der Linux-Datenschredder) umgeleitet werden:

```
./skript > output.txt 2>/dev/null
```

(Unterdrückt die Ausgabe von Fehlern)

Eingabe umleiten

Gleichwohl kann die Standardeingabe (oder Ein- und Ausgabe gleichzeitig) umgeleitet werden

- `cat < meinFile.txt`
- `cat < meinFile.txt > meinKopiertesFile.txt`
- `sort <<fertig`

```
> Z
> B
> A
> fertig
A
B
Z
```

<< fängt eine interaktive Eingabe ab, bis ein Schlüsselwort zur Terminierung eingegeben wird (zB. `fertig`).

Pipeline

Im Gegensatz zu Powershell ist die Pipeline in der Linuxshell nicht Objektorientiert. Es wird lediglich die Ausgabe des vorhergehenden Befehls als textueller Output an den nächsten weitergereicht

- Filtert alle Zeilen mit dem Begriff `hallo` aus der Datei `meinFile.txt`:

  ```
  cat meinFile.txt | grep hallo
  ```

- Filtert und sortiert alle Zeilen mit dem Begriff `hallo` aus der Datei `meinFile.txt` (ohne Duplikate):

  ```
  cat meinFile.txt | grep hallo | uniq | sort
  ```

- liefert eine Liste aller Benutzernamen (Alles vor dem ersten Doppelpunkt in jeder Zeile in `/etc/passwd`), ausser dem Benutzer ntp.

  ```
  cat /etc/passwd | grep -v ntp | cut -d ':' -f 1
  ```

Wildcards

- * steht für beliebig viele Zeichen

```
ls *.txt
```

- ? steht für ein beliebiges Zeichen

```
ls file?.txt
```

- [ ] erzeugt eine Auswahl

```
ls file[123].txt
```

- [-] erzeugt einen Bereich

```
ls file[1-9].txt
```

- ! negiert einen Ausdruck

```
ls file[!3].txt
```

Brace extension

Mit den geschweiften Klammern können Alternativausdrücke formuliert werden:

- erzeugt `File1.txt`, `File2.txt` und `File3.txt`:

  ```
  touch File{1,2,3}.txt
  ```

- Auch Verschachtelungen sind möglich:

  ```
  touch file{orginal{.bak,.txt},kopie{.bak,.txt}}
  ```

  erzeugt `fileoriginal.txt`, `fileoriginal.bak`, `filekopie.txt` und `filekopie.bak`

Tilde expansion

Einige nützliche Erweiterungen der Tilde:

```
Heimverzeichnis des akt. Benutzers: ~

Heimverzeichnis Benutzer: ~BENUTZERNAME

zuvor besuchtes Verzeichnis: ~-

akt. Arbeitsverzeichnis (pwd) : ~+
```