



# Workshop zur Shell-Safari

Bevor wir direkt einsteigen, zwei Dinge: Zum einen ist dies ein **Workshop**, das heißt, **ihr** seid gefragt, aktiv zu werden. Daraus ergibt sich der zweite wichtige Punkt: **Fragt immer, wenn ihr Fragen habt!** Ihr sollt zwar aktiv werden, aber wir sind dazu da, eure Fragen zu beantworten und zu helfen!

## Nun der eigentliche Workshop

Öffne ein Terminal mit **Strg+Alt+t**. Tippe jetzt **pwd** ein. Dieser Befehl sagt dir, in welchem Pfad du dich gerade befindest.

```
pwd  Ausgabe: /home/USER,
```

wobei **USER** für deinen Benutzernamen steht. Jetzt kannst du auch folgende Shell-Befehle eintippen:

```
ls
cd Schreibtisch1
ls
ls -la
pwd
cd ..
```

**Aufgabe:** Versuche zu verstehen, was diese Befehle machen!

**Aufgabe:** Drücke mal die Pfeiltasten **↑** und **↓**. Was passiert? Tippe mal ein **cd Dow** (ohne Enter!) und drücke die Tabulatortaste.

Lade dir den Ordner “Shell\_Safari\_Dateien.zip” unter

[https://redmine.fsmpi.rwth-aachen.de/projects/lip/repository/revisions/master/raw/kiss/workshops/shell/Shell\\_Safari\\_Daten.zip](https://redmine.fsmpi.rwth-aachen.de/projects/lip/repository/revisions/master/raw/kiss/workshops/shell/Shell_Safari_Daten.zip)

herunter. Speichere ihn auf den Desktop (dieser heißt auf deinem Linux wahrscheinlich Schreibtisch).

**Aufgabe:** Springe jetzt mit der Shell in den Desktop (wie?). Gebe dann folgende Befehle ein:

```
unzip Shell_Safari_Daten.zip
```

**Anmerkung:** Sollte eine Fehlermeldung kommen, dass **unzip** nicht installiert ist, tippe in die Konsole: **sudo apt-get install zip unzip**. Damit installierst du die Pakete **zip** und **unzip**. Versuche jetzt noch einmal, die Datei zu entpacken.

Nachdem du den Ordner entpackt hast, springe in das soeben entpackte Verzeichnis mit

```
cd Shell_Safari_Daten
```

und mache ein **ls**. Interessant ist für uns jetzt die Datei **MHL-300K\_17.rso.dat**. Der Name der Datei ist etwas kryptisch, machen wir erst eine Sicherheitskopie:

```
cp MHL-300K_17.rso.dat MHL-300K_17.rso.dat.bak
```

und nennen sie dann in **daten.txt** um:

```
mv MHL-300K_17.rso.dat daten.txt
```

Schaue mit **ls** nach, ob du die Sicherheitskopie und die umbenannte Datei im Ordner findest.

Maximiere jetzt das Terminal, indem du auf das Maximieren-Icon rechts oben klickst. Du kannst dir jetzt die Datei **daten.txt** direkt im Terminal anschauen und musst nicht extra in einen Texteditor wechseln:

```
cat daten.txt
```

Blöd ist jetzt, dass die Datei so lang ist, dass sie nicht mehr auf einen Bildschirm passt. Für diesen Fall gibt es z.B. **more**:

```
more daten.txt
```

Mit Enter kannst du jetzt eine Zeile heruntergehen und mit **q** beendest du **more**.

Du siehst, dass vor der Datei ein Header steht, in dem offensichtlich keine Messdaten stehen. Dieser Header stört uns, wenn wir die Daten später auswerten wollen. Wir wollen ihn also loswerden. Dazu gibt es **head** und

---

<sup>1</sup>Schreibtisch kann auch Desktop oder Arbeitsfläche heißen

`tail. head -5 daten.txt` z.B. gibt dir die ersten 5 Zeilen der Datei an (`tail -5 daten.txt` entsprechend die letzten 5 Zeilen der Datei). Deine nächste **Aufgabe**: Finde mit `head` heraus, wie viele Zeilen Header die Datei hat.

Lösung: -----

Jetzt, wo wir die Anzahl an Header-Zeilen kennen, wollen wir diesen loswerden. Dazu wollen wir erst einmal die Datei ohne Header einfach nur im Terminal ausgeben. Wüssten wir, wie viele Zeilen die Datei insgesamt hat, könnten wir das mit `tail` die Datei ohne Header anzeigen lassen (**Aufgabe**: wie?). Glücklicherweise gibt es so ein Programm, nämlich: `wc -l` (`wc` = wordcount, `l` = lines):

```
wc -l daten.txt
```

Lösung, um die Datei headerfrei anzuzeigen: -----

Jetzt ist es blöd, dass wir die headerfreie Datei nur im Terminal angezeigt bekommen; viel schöner wäre es, das Angezeigte in eine Datei zu schreiben. Dazu gibt es z.B. die Umleitung `>`. **BEFEHLE** `> daten1.txt` schreibt die Ausgabe von Befehle in die Datei `daten1.txt`, die auch direkt erzeugt wird, falls sie noch nicht existiert. **Aufgabe**: Erzeuge eine headerfreie Datei mit Namen `daten1.txt`:

Lösung: -----

Wie du schon gemerkt hast, hat die Datei sehr viele Spalten. Wird sind aber nur an der 3. und 6. Spalte interessiert. Um nur diese Spalten anzuzeigen gibt es den Befehl `cut`. Dazu müssen wir `cut` sagen, mit welchem Trennzeichen die Spalten getrennt sind und welche Spalten er uns anzeigen soll. Dies machen wir mit:

```
cut -d "TRENnzeichen" -f SPALTE1,SPALTE2 daten1.txt
```

**Aufgabe**: Lasse dir nur die 3. und 6. Spalte von `daten1.txt` anzeigen. Modifiziere den Befehl dann so, dass das Ergebnis nur in die neue Datei `daten2.txt` geschrieben wird.

Lösung: -----

Jetzt stört uns noch, dass beide Spalten mit einem Komma getrennt sind. Wir wollen daher alle Kommata durch ein Leerzeichen ersetzen. Dazu gibt es selbstverständlich auch das Programm: `tr`.

```
tr ", " " "
```

würde z.B. alle Kommata durch Leerzeichen ersetzen. Blöd nur, dass man hinter dieses `tr` keinen Dateinamen schreiben kann. Es ist fast so, als würde `tr` darauf warten, dass es etwas von einem Programm bekommt. Jetzt wissen wir, dass wir mit `>` Programmausgaben in Dateien umlenken können. Nur müssen wir statt in eine Datei jetzt eine Ausgabe aber in ein anderes Programm, nämlich nach `tr` umleiten. Dafür gibt es auch einen Operator, nämlich die Pipe `|`. Ihr wisst, dass `cat daten2.txt` euch den Inhalt von `daten2.txt` anzeigt. Ihr wisst jetzt auch, dass die Pipe `|` diese Ausgabe weiterleitet (etwa an `tr`). **Aufgabe**: Verbindet nun dieses Wissen, um euch den Inhalt von `daten2.txt` mit Leerzeichen statt Kommata anzeigen zu lassen:

Lösung: -----

Diese Ausgabe wiederum will man jetzt in die Datei `daten3.txt` schreiben. Dazu müsst ihr hinter den letzten Befehl einfach `> daten3.txt` schreiben:

Lösung: -----

Jetzt, wo ihr auch die Pipe `|` kennt, könnt ihr alle Befehle sogar so verbinden, dass ihr euch die Dateien `daten1.txt` und `daten2.txt` sparen könnt und direkt die headerfreie Datei mit nur der 3. und 6. Spalte ohne Kommata in die Datei `daten4.txt` schreiben könnt. Dazu müsst ihr alle Befehle über die Pipe `|` verbinden. **Aufgabe**: Macht genau das! Tipp: Wenn ihr einen Dateinamen hinter den Befehl geschrieben habt, lasst ihr den Dateinamen einfach weg, wenn der Befehl auf eine Pipe `|` folgt.

Lösung: -----

## Wer noch Lust hat...

Es gibt noch mehr Dateien mit dem Namen MHL-300K.... Jetzt kann man natürlich für jede dieser Dateien obige Prozedur wiederholen. Viel spannender wäre es aber, ein **Skript** zu schreiben, was das automatisch für alle Dateien macht. Nimm dazu deinen Handout-Zettel zur Hand (Abschnitt **Skripte erstellen** und **if und for im Skript**) und versuche es einmal selbst! Hier gilt natürlich umso mehr: Melde dich, wenn du Fragen habt! ☺