# Übung Unix-Kommandozeile

*Patrick Bucher, seantis GmbH*

03.09.2020

Die Kommandozeile ist zwar eine recht alte, aber sehr effektive Methode um einen Computer zu bedienen. Server ohne grafische Benutzeroberfläche können nur über die Kommandozeile bedient werden.

Dieses Dokument stellt die Kommandozeile von Unix vor, die u.a. auf Linux und macOS zum Einsatz kommt. Es gibt verschiedene Unix-Kommandozeilen, die folgenden Erklärungen sollten aber für die meisten gelten.

## Form der Befehle

Die Befehle auf der Kommandozeile haben die folgende Form:

```
$ Befehl Parameter1 Parameter2 ... ParameterN
```

Das Dollarzeichen muss nicht eingegeben werden, denn es ist Teil der Eingabeaufforderung.

Der Befehl `echo` gibt alle angegebenen Parameter auf den Bildschirm aus:

```
$ echo Das ist ein Test
Das ist ein Test
```

Der `echo`-Befehl gibt hier *vier* Parameter aus. Mehrere Parameter können mit Anführungs- und Schlusszeichen zusammengefasst werden:

```
$ echo 'Das ist' "ein Test"
Das ist ein Test
```

Die Ausgabe sieht immer noch gleich aus, aber `echo` hat nur noch zwei Parameter erhalten.

Neben den Parametern unterstützen die einzelne Befehle auch Optionen. Mit der Option `-n` von `echo` verhindert man, dass am Ende der Ausgabe ein Zeilenumbruch geschrieben wird:

```
$ echo -n 'Das ist ein Test'
Das ist ein Test$
```

(Die neue Eingabeaufforderung erscheint dadurch auf der gleichen Zeile wie die Ausgabe).

Verschiedene Befehle unterstützen verschiedene Optionen. Es gibt Optionen in der Kurzform (`-n`) und in der Langform (`--version`). Bei manchen Befehlen gibt es für die gleiche Option jeweils eine kurze *und* eine lange Form.

### Weiterleitung

Die Ausgabe eines Befehls kann mit dem Operator > in eine Datei geschrieben werden:

```
$ echo 'Das ist ein Test' > output.txt
```

Mit dem Befehl `wc` (*word count*) können die Anzahl Zeilen, Wörter und Zeichen gezählt werden. Mithilfe des Operators < kann die Eingabe für diesen Befehl aus einer Datei gelesen werden:

```
$ wc < output.txt
1       4      17
```

Diese Ausgabe bedeutet, dass die Datei `output.txt` eine Zeile, vier Wörter und siebzehn Zeichen enthalte.

Mithilfe einer sogenannten *Pipe* (engl. für "Rohr") kann die Ausgabe von einem Befehl in die Eingabe eines anderen Befehls umgeleitet werden. Eine Pipe wird mit dem Operator | umgesetzt (auf der Tastatur `[Alt Gr] + [7]`). Hier wird `wc` direkt auf die Ausgabe von `echo` angewendet, ohne wie vorher den Umweg über eine Datei zu nehmen:

```
$ echo 'Das ist ein Test' | wc
1       4      17
```

### Wichtige Befehle

Auf einer typischen Installation gibt es hunderte oder tausende von Befehlen. Einige grundlegende Befehle werden hier mit einigen Optionen kurz erläutert:

#### `cat` - Dateien aneinanderhängen und in die Standardausgabe schreiben

```
-n, --number
    alle Ausgabezeilen nummerieren
```

#### `cd` - in ein anderes Arbeitsverzeichnis wechseln

```
..
    in das übergeordnete Arbeitsverzeichnis wechseln
```

#### `du` - gibt den Platzverbrauch von Dateien an

```
-b, --bytes
    Anzahl bytes ausgeben
-c, --total
    Gesamtsumme erstellen
-h, --human-readable
    Größen in menschenlesbarem Format ausgeben (z. B. 1K 234M 2G)
```

#### `head` - den ersten Teil von Dateien ausgeben

```
-n K
    die ersten K Zeilen statt der ersten 10 ausgeben
```

#### `ls` - Verzeichnisinhalte auflisten

```
-a, --all
    Einträge nicht ignorieren, die mit ».« beginnen
-l
    Langes Listenformat verwenden
```

**`pwd` - den Namen des aktuellen Arbeitsverzeichnisses ausgeben**

**`sort` - Zeilen von Textdateien sortieren**

```
-f, --ignore-case
    Klein- als Großbuchstaben behandeln
-n, --numeric-sort
    Anhand des numerischen Werts der Zeichenkette vergleichen
-r, --reverse
    Das Ergebnis der Vergleiche umkehren
```

**`tail` - den letzten Teil von Dateien ausgeben**

```
-n K
    die letzten K Zeilen ausgeben statt der letzen 10
```

**`uniq` - aufeinanderfolgende doppelte Zeilen berichten oder entfernen**

```
-c, --count
    Den Zeilen die Häufigkeit des Vorkommens voranstellen
-i, --ignore-case
    Beim Vergleichen nicht auf Groß- und Kleinschreibung achten
```

**`wc` - Anzahl der Zeilen, Wörter und Byte für jede Datei ausgeben**

```
-c, --bytes
    Anzahl der Bytes ausgeben
-m, --chars
    Anzahl der Zeichen ausgeben
-l, --lines
    Anzahl der Zeilen ausgeben
-w, --words
    Anzahl der Wörter ausgeben
```

## Aufgaben

Mit den folgenden Aufgaben kannst Du die Bedienung der Kommandozeile etwas üben. Die vorherige Abschnitte sollten alle Informationen enthalten, um die Aufgaben zu lösen. Manche sind einfacher, manche schwieriger, und manche gar nur mit Vorwissen zu lösen.

Wenn etwas nicht funktioniert, oder Du bei einer Aufgabe fünf Minuten lang nicht weiter kommst, frage einfach den Instruktor.

Halte die Lösungen in einer separaten Textdatei fest. (Speichere diese häufig ab, um keine Informationen zu verlieren.) Obwohl man eine Textdatei nicht wie ein Word-Dokument formatieren kann, sollst Du diese möglichst gut lesbar formatieren. Bewahre diese Datei bis zum Schluss auf!

Um mit den Aufgaben beginnen zu können, musst Du im richtigen Arbeitsverzeichnis sein. Der Instruktor wird Dir dabei behilflich sein.

Bei vielen Aufgaben werden mehrere Befehle benötigt. Diese kannst du mittels der weiter vorne beschriebenen *Pipe* kombinieren.

**Aufgabe 0: Befehle ausprobieren**

Probiere die verschiedenen Befehle, die weiter oben aufgelistet sind, einmal aus. Probiere auch verschiedene Optionen aus. Versuche auch, einige Befehle mittels Pipe zu kombinieren. Halte einige Beispiele hierzu fest.

**Aufgabe 1: Dateien anzeigen**

Welche Dateien befinden sich im aktuellen Arbeitsverzeichnis?

**Aufgabe 2: Dateien nach Muster anzeigen**

Die Kommandozeile ergänzt Ausdrücke der Form `*.xyz` automatisch mit allen Dateinamen, welche die Endung `.xyz` haben.

Welche Dateien mit der Endung `.txt` befinden sich im aktuellen Arbeitsverzeichnis?

**Aufgabe 3: Zeilen zählen**

Wie viele Zeilen enthalten die beiden Textdateien, die Du in Aufgabe 2 gefunden hast?

**Aufgabe 4: Zeichen zählen**

Wie viele Zeichen enthalten die beiden Textdateien?

**Aufgabe 5: Bytes zählen**

Wie viele Bytes enthalten die beiden Textdateien?

Kann man das mit verschiedenen Programmen herausfinden?

Unterscheidet sich die Anzahl Bytes von der Anzahl Zeichen? Warum bzw. warum nicht?

**Aufgabe 6: Eindeutige Einträge anzeigen**

Die Dateien `search-small.txt` und `search-big.txt` enthalten verschiedene Lebensmittel. Dabei kommen die gleichen Lebensmittel mehrmals vor.

Löse die folgenden Aufgaben jeweils für *beide* Dateien.

Welche Lebensmittel kommen in den beiden Listen vor?

Tipp: Verwende den `uniq`-Befehl. Warum müssen die Daten zunächst sortiert werden, damit das richtig funktioniert?

**Aufgabe 7: Anzahl verschiedener Einträge zählen**

Auf Basis von Aufgabe 6 ist nun bekannt, welche Lebensmittel in den Dateien auftauchen.

Wie kann die Anzahl der Lebensmittel automatisch ermittelt werden?

**Aufgabe 8: Anzahl der Vorkommnisse zählen**

Wie oft tauchen die verschiedenen Lebensmittel in den beiden Textdateien auf?

Tipp: Du hast den Befehl schon weiter oben verwendet, benötigst aber einen zusätzlichen Parameter zum Zählen.

**Aufgabe 9: Bestenliste**

Auf Basis der Liste von Aufgabe 8 soll eine Bestenliste erstellt werden. Je häufiger ein Eintrag in einer Datei vorkommt, desto weiter oben soll er stehen.

Tipp: Du musst den gleichen Befehl zweimal verwenden, jedoch mit anderen Optionen.

**Aufgabe 10: Top Ten**

Die Bestenliste von Aufgabe 9 soll verkürzt werden, sodass nur die zehn obersten Einträge ausgegeben werden. Speichere die Bestenlisten in den Dateien `top-ten-small.txt` und `top-ten-big.txt` ab.

**Aufgabe 11: Seltenste Einträge**

Wie lässt sich analog zu Aufgabe 9 und 10 eine Liste mit den zehn seltesten Einträgen erstellen? Speichere die beiden Ergebnisslisten in den Dateien `bottom-ten-small.txt` und `bottom-ten-big.txt` ab. Kannst Du verschiedene Lösungen finden?

**Aufgabe 12: Was soll das alles?**

In den letzten 11 Aufgaben hast Du verschiedene Probleme gelöst und dabei mehr oder weniger interessante Informationen ermittelt.

Überlege Dir zu jeder Aufgabe, was diese für einen Praxisbezug haben könnte. Stelle Dir dazu bei jeweils die Frage: "Wozu könnten die ermittelten Informationen nützlich sein?" Notiere deine Ideen, wir werden diese anschliessend besprechen.

**Aufgabe 13: Wie findest Du das?**

Arbeitest Du gerne auf der Kommandozeile? Oder findest du die Bedienung mühsam? Warum arbeitet man überhaupt auf der Kommandozeile und nicht auf einer grafischen Oberfläche? Welche Vor- und Nachteile hat die Kommandozeile gegenüber grafischen Oberflächen?

**Aufgabe 14: Weitere Ideen?**

Welche Sachen könnte man noch auf der Kommandozeile machen? Hast Du auf Deinem eigenen Computer zu Hause auch Sachen, die man mit der Kommandozeile herausfinden könnte?

**Aufgabe 15: Weitere Dateien**

Vielleicht ist Dir aufgefallen, dass es im Arbeitsverzeichnis noch andere Dateien als `search-big.txt` und `search-small.txt` gibt.

Überlege Dir zu diesen Dateien folgende Fragen (stichwortartige Notizen genügen):

- Was sind das für Dateien?
- Kannst Du Dir vorstellen, wozu diese gut sind?
- Kommt Dir etwas in diesen Dateien bekannt vor?
- Gibt es einen Zusammenhang zwischen den Dateien in diesem Verzeichnis, oder sind diese einfach nur zufällig da?
- Mit welchem Programm kannst Du diese Dateien öffnen?
- Kannst Du auch Befehle auf diese anwenden?