Linux und Textdateien

Effizient arbeiten mit einfachen Mitteln

Patrick Bucher

26.04.2017

Windows- und Mac-User schreiben Texte meistens mit *Word*. Mit *LibreOffice Writer* gibt es eine kostenlose Alternative zu *Word*, die auch unter Linux läuft. Hartgesottene Linux-User verwenden aber lieber einen Texteditor zum Schreiben ihrer Texte. Warum ist das so?

Microsofts kostenpflichtige Textverarbeitung *Word* aus der *Office*-Suite verwendet als Dateiformat *Office Open XML* (OOXML). Die Open-Source-Textverarbeitung *LibreOffice Writer* verwendet hingegen *OpenDocument Text* (ODT) als Dateiformat. Beide Programme (*Word* und *Writer*) unterstützen beide Formate (OOXML und ODT).

Texteditoren verwenden als Dateiformat einfache Textdateien.

Unterschiede zwischen Textdateien und *OOXML/ODT*

OOXML und ODT speichern Texte in einer komprimierten XML-Struktur ab. Einfache Textdateien haben keine solche Struktur, sondern bestehen nur aus einer Reihe von Zeichen. Dadurch ergeben sich einige Unterschiede in der Handhabung von OOXML/ODT-Dateien einerseits und Textdateien andererseits:

1. OOXML/ODT-Dateien lassen sich nur mit *Word* und *Writer* zuverlässig bearbeiten. Textdateien lassen sich mit einem beliebigen Texteditor bearbeiten.
2. OOXML/ODT-Dateien enthalten viele Zusatzinformationen für Formatierung, Struktur und zusätzliche Einstellungen. Textdateien haben diesen *Overhead* nicht, da sie nur die eigentlichen Nutzdaten enthalten.
3. OOXML/ODT-Dateien können nur so lange gelesen werden, wie die entsprechenden Programme dazu verfügbar sind. Textdateien, die nur aus einer Reihe von Zeichen bestehen, können immer gelesen werden.
4. OOXML/ODT-Dateien lassen sich nur im «WYSIWYG»-Modus («what you see is what you get») bearbeiten. Textdateien lassen sich auch als *Textstrom* bearbeiten.

Die ersten drei Punkte sprechen klar *für* den Einsatz von Textdateien. Doch welche Vorteile bietet ein *Textstrom*?

Beispiel: Wörter in Artikeln zählen

Angenommen wir haben eine Reihe von Artikeln; einmal im ODT-Format (Endung `.odt`) und einmal im Textformat (Endung `.txt`). Nun wollen wir herausfinden, welcher Artikel am meisten Wörter enthält. Mit unseren ODT-Dateien verfahren wir folgendermassen:

1. Wir öffnen den ersten Artikel mit *Writer*.
2. Wir gehen auf das Menü *Tools* und wählen den Eintrag *Word Count*.
3. Wir notieren uns den Dateinamen und die Anzahl der Wörter dazu.
4. Wir wiederholen den Vorgang für den nächsten Artikel.

Dieses Vorgehen ist sehr aufwändig. Zudem muss der Vorgang später wiederholt werden, wenn die Artikel in der Zwischenzeit verändert worden sind.

Der `wc`-Befehl

Mit Textdateien sieht das Vorgehen anders aus. Wir öffnen ein *Terminal* und verwenden den `wc`-Befehl:

```
$ wc -w *.txt
```

- `wc` steht für «word count» und zählt die Wörter in einer Datei.
- Standardmässig gibt `wc` die Anzahl Zeilen, Wörter und Zeichen einer Datei aus. Mit dem Parameter `-w` (für «words») erhalten wir nur die Anzahl Wörter.
- Mit `*.txt` übergeben wir dem Programm sämtliche Textdateien im Arbeitsverzeichnis.

Dadurch erhalten wir die folgende Ausgabe:

```
1739 berlinreise.txt
2220 eigenes-bier-brauen.txt
 893 im-stau.txt
1231 neues-aquarium.txt
```

Die Ausgabe an `sort` weiterleiten

Wir sehen die Anzahl Wörter in der ersten Spalte, doch die Zeilen sind nicht richtig, d.h. nach der Anzahl der Wörter sortiert. Über das Muster `*.txt` erhält `wc` die Dateien in alphabetischer Reihenfolge – und gibt sie auch in alphabetischer Reihenfolge wieder aus. Darum verwenden wir zusätzlich den `sort`-Befehl.

```
$ wc -w *.txt | sort -n -r
```

- Zwischen den Befehlen steht das Zeichen `|`. Das ist eine sogenannte *Pipe*, zu Deutsch etwa «Röhre». Eine Pipe nimmt die Ausgabe eines Programmes entgegen und leitet sie

als Eingabe zum nächsten Programm weiter. **Der Text «fliesst» als *Textstrom* durch die Röhre.**
- Es folgt der `sort`-Aufruf, wodurch Textzeilen in alphabetisch aufsteigender Reihenfolge sortiert werden.
- Da wir keine alphabetische, sondern eine numerische Sortierung brauchen («100» wäre alphabetisch sortiert kleiner als «9»), geben wir den Parameter `-n` (für «numeric») an.
- Zudem soll die Reihenfolge nicht aufsteigend (die kleinste Zahl zuerst) sondern absteigend (die grösste Zahl zuerst) sein, was wir mit dem Parameter `-r` (für «reverse») erreichen.

Die Ausgabe sieht nun so aus:

```
2220 eigenes-bier-brauen.txt
1739 berlinreise.txt
1231 neues-aquarium.txt
 893 im-stau.txt
```

Der Artikel mit den meisten Wörtern ist `eigenes-bier-brauen.txt`. Für vier Dateien mag das nicht besonders beeindruckend sein. Hätten wir tausende von Dateien, müssten wir zwar die Ausgabe kürzen, ansonsten wäre aber das Vorgehen genau gleich. Doch die Zeitersparnis gegenüber dem manuellen Vorgehen mit ODT-Dateien gewaltig. **Die Lösung mit den Textdateien *skaliert* wesentlich besser.**

Das ist der Grund, warum viele Linux-User Textdateien und die Kommandozeile mit ihren Befehlen grossen Softwarelösungen gegenüber bevorzugen. **Ein Problem muss nur einmal gelöst werden. Die Lösung kann immer wieder auf ähnliche Probleme angewandt werden.**

Abbildung 1 veranschaulicht die Befehlszeile und den Ablauf:

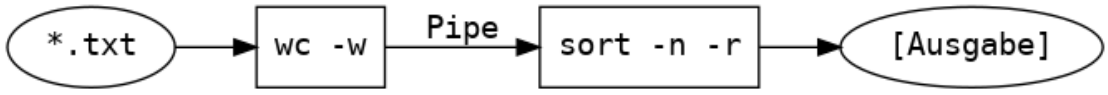

Abbildung 1: Die Befehlszeile veranschaulicht

Und jetzt?

Wir haben gesehen, dass sich ein alltägliches Problem mithilfe von Textdateien, einfachen Befehlen (wie `wc` und `sort`) und Textströmen effizienter lösen lässt als mit einer Textverarbeitung.

Wer noch weitere solche Techniken kennenlernen möchte, dem empfehle ich den Vortrag *Power Use of UNIX* von Dan North.