Library Caprentry
Konrad Förstner und Till Sauerwein
Workshop April 2019

Notizen von Dina Heß

2. April 2019

Inhaltsverzeichnis

1 Library Carpentry - Begriff

Der Begriff Library Carpentry kommt von Software Carpentry. Dabei war die Idee, WissenschaftlerInnen spezifische IT-Skills für ihren Bereich an die Hand zu geben. Dabei ist wichtig, dass alle Vorgänge den folgenden Grundsätzen genügen. Sie sind

- Reproduzierbar
- Transparent
- Nachvollziehbar (Tracking)

Aus Software Carpentry entwickelte sich Data Carpentry und verschmolz somit zu *The Carpentries*. Schließlich entstand daraus 2014 auch Library Carpentry.

Der Kurs soll als Startpunkt für Library Carpentry gesehen werden. Austausch ist wichtig! Anregung (Konrads Team): wöchentliche HackieHour. Ziel ist:

- Automatisierung, Optimierung von (wiederholbaren) Tasks und Prozessen
- Reproduzierbarkeit und Transparenz (nicht immer das Rad neu erfinden, nachnutzbar, nachvollziehbar)
- Gibt größere Handlungsfähigkeit, größeren Handlungsspielraum (warum eine teure Software kaufen, wenn man Software mit demselben Ziel mit offenen Methoden selbst und offen *nachnutzbar* selbst machen kann – kann Geld sparen!)

2 Unix-Shell

Alternatives (zu anderen, vgl. Mac Terminal) Interface/Schnittstelle zum Betriebssystem. Wir nutzen die Version *Bash* einer solchen Shell. Die Windows-Version davon heißt Git bash.

> Die Shell ist mächtig.
> Sie kann zum Kopieren, Verschieben und Kombinieren mehrerer Dateien (auch gleichzeitig und in großer Menge) verwendet werden.

2.1 Erste Schritte

Zu unterscheiden zwischen den primären Befehlen (nach dem $-Zeichen), Argumenten (Flags) und Parametern.
⋆ Hier handschriftliche Notizen einfügen

`ls -help` listet mögliche Befehle/Eingaben und allgemeine Hilfe.
`mkdir firstdir`. Mit `TAB` kann man Befehle bzw. Parameter auto-vervollständigen. `ls -a` listet nun alles auf, was da ist. In einem leeren Ordner stehen da die Optionen `./` bezeichnet das aktuelle Verzeichnis, `../` bezeichnet das darüberliegende Verzeichnis

(Eltern-Ordner).

Faulheit ist eine Tugend!

Um den Großeltern aufzurufen ist auch der Befehl `cd ../..` möglich.

Es handelt sich hierbei um relative Pfade. `$ cd /c/Users/Dina/Desktop/Library Carpentry 2019/shell-lesson` ist dabei der absolute Pfad. Um dabei ins Home-Directory zu kommen nutzt man die Tilde, bzw. ein simples `tilde` bringt mich ins Home-Directory.

Mit den Pfeil-hoch und Pfeil-runter Tasten kann man vorher genutzte Befehle aufrufen und ggf. erneut verwenden.

Durch `STRG+r` kann ich ein Muster eingeben, um ähnliche Befehle wieder aufzurufen (nacheinander wieder mit `STRG+r`. Mit `STRG+c` kann man das wieder abbrechen – auch Programme stoppen).

Mit `STRG+a` kann man im Befehl an den Anfang, mit `STRG+e` ans Ende und mit `STRG+k` nach dem Cursor löschen, mit `STRG+u` alles vor dem Cursor löschen.

2.1.1 Dateiendungen betrachten

`json` – JavaScript-Ding (.js ausführbar). Alle in shell-lesson befindlichen Dateien sind letztendlich Textdateien (Plaintext), die man mit Texteditoren bearbeitet werden können.

2.2 Plaintext-Dateien lesen

Der Befehl `cat` gibt den Text einer Plaintext-Datei aus und gibt es als Rückgabe zurück. `head` gibt nur die ersten 10 Zeilen aus. Äquivalent dazu der Befehl `tail` mit den letzten 10 Zeilen.
`head -n 20 name.txt` gibt die ersten 20 Zeilen aus.

Mit dem Befehl `less` können Daten strukturiert angezeigt werden (mit `q` kommt man wieder raus). Mit `less -S datei.tsv` kann man das auch noch als Tabelle anzeigen lassen.

Fazit: Wir können die Dateien anschauen, sie ausgeben, Teile ausgeben und Dateien strukturiert ausgeben. Mit `head -n 5 datei1.txt datei2.txt` kann man auch zwei ausgeben (erste 5 Zeilen hier). Nervig wirds so bei mehr Dateien. `head -n h *txt`. Dabei steht der * repräsentantiv für mehrere Dinge. Dies nennt man auch Globbing (wiederholbar für weitere solche Dinge). Es können alle möglichen Teile auch kombiniert werden: `ls 2014*31*tsv`. Ähnlich zu etwa der Arbeit mit regulären Ausdrücken.

Mit `ls -1` kann man übrigens die Liste als 1 Spalte ausgeben.

2.3 Konvention

Datumsangaben immer 2019-04-02 (ISO 8601), inkl. Bindestriche (darf man weglassen, sollte man aber nicht). Überlegt auch immer, wie man die Dateien nennt.

2.4 Dateien verschieben und umbenennen etc.

`mkdir dateiname` – Verzeichnis erstellen
`mv dateiname.txt neuername.txt` – Datei umbenennen. Beachte hier: kontextabhängiges Verhalten:
`mv dateiname.txt verzeichnisname/` – verschiebt in ein anderes Verzeichnis.
`mv dateiname.txt firstdir/neuername.txt` – verschieben und umbenennen.
`mv anderesverzeichnis/dateiname.txt .` – zurückschieben ins aktuelle Verzeichnis.
`cp dateiname.txt neuedatei.txt` – erstellt eine Kopie der ersten Datei mit dem Namen neuedatei.txt

2.5 Datenformate

Betrachten mal .html oder .json .tsv (Tabellen) usw.

2.6 For-Schleifen/For-loops

Konzept, das in allen Programmiersprachen zu finden ist. Prinzip: wende eine Aktivität auf mehrere Dinge an.

`touch` ändert Zeitstempel oder erstellt Datei (falls noch nicht existiert). Angenommen, wir wollen einen Befehl wie `cp` mehrfach anwenden auf alle Dateien, die einem Muster genügen. → For-Schleife.

Syntax einer For-Schleife

```
FOR...IN Menge
DO...
done.
```

Im **Beispiel** also:

```
for FILE in {a.pdf, b.pdf, c.pdf}
do
cp $ FILE $ FILE-Backup
done
```

Beachte: In FILE werden hier die Dateinamen abgelegt. Ferner ist hier anstelle einer Menge in Unix-Shell einfach eine Liste anzugeben.

Dabei ist `echo` ein Befehl zur Ausgabe (besser: `printf`)

```
Dina@LAPTOP-CR9NL23A MINGW64 ~/Desktop/Library Carpentry 2019/shell-lesson
$ for FILE in a.pdf b.pdf c.pdf ; do  echo $FILE ist eine schöne Datei; cp $FILE ${FILE}-backup; done
a.pdf ist eine schöne Datei
b.pdf ist eine schöne Datei
c.pdf ist eine schöne Datei
```

Abbildung 1:

2.6.1 Schöne Befehle

`wc` – Wordcount
`wc -l` – Anzahl Zeilen
`rm` – löscht eine Datei
`rm -i *txt` – entfernt alle txt-Dateien, fragt aber vorher.

```
$ wc *tsv
  507732  17606310 131122144 2014-01_JA.tsv
   13712    511261   3773660 2014-01-31_JA-africa.tsv
   27392   1049601   7731914 2014-01-31_JA-america.tsv
    5375    196999   1453418 2014-02-02_JA-britain.tsv
  554211  19364171 144081136 total
```

Abbildung 2:

Eine Datei mit den Ausgaben erstellen geht durch `>`: siehe Abb. 3

```
$ wc -l *tsv > lengths.txt
```

Abbildung 3:

Da die Länge an erster Stelle steht (als Zahl), kann ich die ausgegebene Liste sortieren nach Länge. Siehe Abb. 4

Man kann dann durch `$ head -n 1 sorted-lengths.txt` die Datei mit der kürzesten Länge angeben.

2.6.2 Pipes

Das Zeichen `|` sagt: der Output aus links wird direkt in rechts reingesteckt (Abb. 5).

Beispiel

Ich kann also einen gescannten Text erst in eine OCR-Software, das Ergebnis in eine Texttdatei schreiben und diese durch ein Indexierungsprogramm jagen. Mit einem Befehl.

Abbildung 4:

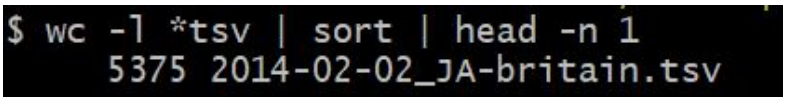

Abbildung 5:

Beachte: Dateien werden mit `>` einfach überschrieben. Nicht aber mit `»`.

`grep -c` mit count. nicht nur Ausgabe.

Man kann mit einer Handvoll weniger Befehle schöne Sachen zusammen bauen. Man kann solche einfachen Analysen machen oder Dinge automatisieren. Ich wende Aktionen auf viele Dateien an. Zum Thema Reproduzierbarkeit: Skripte.

2.7 Shell-Skripte

Auch Shell-Skripte sind wiederum Plaintext-Dateien, die eine Liste von Befehlen enthält. Eine Art *Kochrezept zum Ausführen*. Arbeite mit dem Editor nano. Dieser öffnet/generiert eine Datei namens count_lines.sh (siehe Abbildungen 7f.). Damit wir erkennen, dass es ein Shell-Skript ist, enden wir mit .sh. In den gängigen Editoren spricht man von Syntax-Highlighting (Färben bestimmter Wörter und Zeichen nach ihrer Funktion). (Nano-Befehle: Strg+o zum Speichern, Strg+x um den Editor zu verlassen). Beachte nun: Mit `$1` kann ich Variablen einlesen (vorher stand da `*txt`).

⇒ Programmierung mit Variablen! Morgen können wir die Änderungen (hier von Einfach zu Komplexer) mit Git tracken.

3 Python

Python ist eine der am stärksten wachsenden und populärsten Sprachen. Sie ist vergleichsweise einfach zu lernen und sehr mächtig. Python ist zudem frei verfügbar und hat inzwischen eine sehr große Community.

Merke: Jupyter Notebook. Könnte die Zukunft des Wissenschafts-Outputs sein (statt

Abbildung 6:

klassische Aufsätze), da man Code, Kommentar und ausgeführten Code (z.B. Bilder) kombinieren kann. Durch # kann auch hier ein Kommentar eingesetzt sein.

False und True als Boolesche Variablen

Container anlegen `variablenname` $= [1, 2, 3]$. Die Zahl an der ersten Stelle aufrufen kann man dann durch `variablenname`[0].Beachte, um das zu tun, die erste Zeile vorher *ausgeführt* werden muss. Funktion: `len` für die Länge eines Containers/einer Variable. (Bei Help – Keyboard Shortcuts gibrs ein paar Befehle für die Faulen).

Mit `variablenname.upper()` wird der String darin in Großbuchstaben ausgegeben. Analog mit `lower` klein. Mit ? lassen sich Infos zum jeweiligen Objekt anzeigen.

3.1 For-Schleifen

Vergleiche ab hier auch das 1. Jupyter-Notebook

3.2 Built-in-functions

```
print()
len()
```

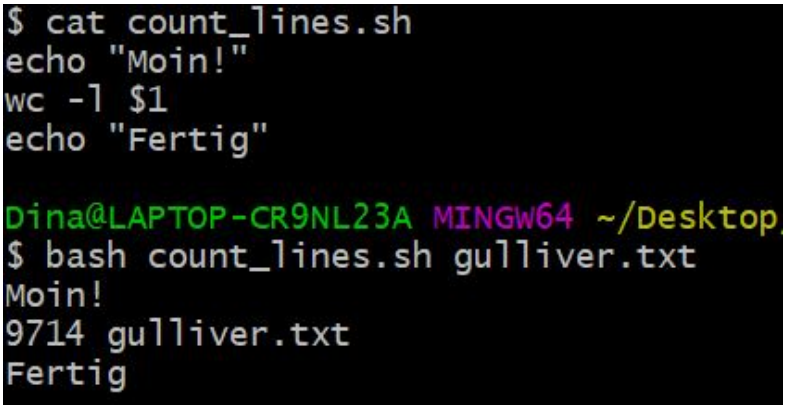
$ cat count_lines.sh
echo "Moin!"
wc -l $1
echo "Fertig"
Dina@LAPTOP-CR9NL23A MINGW64 ~/Desktop
$ bash count_lines.sh gulliver.txt
Moin!
9714 gulliver.txt
Fertig

Abbildung 7: