Pandoc Power-Features
(Vorab-Version)

Kurt Pfeifle <kurt.pfeifle@mykolab.com>

FrOSCon 2015,
St. Augustin,
22. August 2015

Dieses Dokument baut auf der Einführung in das Textformat *Markdown* auf, welche im Artikel *"Markdown und 'ne Prise pandoc"* (siehe Link unter "Files" auf der FrOSCon15-Webseite zum Workshop)[1] zu finden ist.

Es vertieft die dort erfolgte Darstellung des Tabellen-Codes und stellt insbesondere vor, wie man mittels `pandoc` aus Markdown-Quellen Ausgabe-Dokumente in LaTeX, PDF, HTML, EPUB, ODT oder DOCX (MS Word) erzeugt, welche automatisch erzeugte Literatur-Verzeichnisse beinhalten. Daher ist es empfehlenswert, jenen Beitrag zuerst zu lesen.

Das Dokument ist insofern selbst-referenzierend, als es zum Teil seinen eigenen Quellcode dokumentiert **;-)**

Selbstverständlich ist sein Quelltext in Markdown geschrieben! Und ebenso selbstverständlich hat `pandoc` das dem Leser jetzt vorliegende Dokumenten-Format aus genau diesem Markdown erzeugt – was immer dieses Format auch sei: PDF, HTML, EPUB, DOCX oder ODT.

Das Große Einmaleins von Markdown

Wiederholung und tabellarische Übersicht

Die folgende Übersicht stellt nochmals die wichtigsten Elemente von Markdown-Quellcode und dessen Darstellung im Ausgabe-PDF gegenüber, welches `pandoc` daraus erzeugen kann:

[1] http://programm.froscon.de/2015/events/1620.html

Markdown-Quelltext (Input)	**Ergebnis–PDF** (Output)
`*kursiv*`	*kursiv*
`**fett**`	**fett**
`***fett+kursiv***`	***fett+kursiv***
`inline `code` Wörter`	inline `code` Wörter
`` `~~duRchgesTrichen~~` ``	~~duRchgesTrichen~~
`Einfügen eines [Hyperlink zur FrOSCon15](http://goo.gl/BnsF0G)`	Einfügen eines Hyperlink zur FrOSCon15
`Absätze: Zwei Absätze werden durch eine Leerzeile voneinander getrennt. Zeilenschaltungen ohne Leerzeile bewirken keinen neuen Absatz.`	Absätze: Zwei Absätze werden durch eine Leerzeile voneinander getrennt. Zeilenschaltungen ohne Leerzeile bewirken keinen neuen Absatz.

Markdown-Quelltext (Input)	**Ergebnis–PDF** (Output)

`Un-nummerierte Liste:`
 	
`* alpha`
`* beta`
`* gamma`
	Un-nummerierte Liste:
	
	• alpha
	• beta
	• gamma

`Un-nummerierte Liste (alternativ mit '+'-Zeichen statt '*'):`
 	
`+ alpha`
`+ beta`
`+ gamma`
	Un-nummerierte Liste (alternativ mit '+'-Zeichen statt '*'):
	
	• alpha
	• beta
	• gamma

`Un-nummerierte Liste (alternativ mit '-'-Zeichen statt '*'):`
 	
`- alpha`
`- beta`
`- gamma`
	Un-nummerierte Liste (alternativ mit '-'-Zeichen statt '*'):
	
	• alpha
	• beta
	• gamma

`Nummerierte Liste:`
`1. eins`
`1. zwei`
`1. drei`
	Nummerierte Liste:
	
	1. eins
	2. zwei
	3. drei

`# Überschrift 1. Ordnung`	**Überschrift 1. Ordnung**
`## Überschrift 2. Ordnung`	**Überschrift 2. Ordnung**

Markdown-Quelltext (Input)

```
### Überschrift 3. Ordnung


Nummerierte Liste, geht auch so
(ist aber mehr Arbeit, sollte
mal ein Umsortieren erforderlich
werden):

1. eins
2. zwei
3. drei


Nummerierte Liste mit Offset:

7. eins
8. zwei
8. drei
8. vier


Nummerierte Liste mit Offset:

(7) eins
(8) zwei
(8) drei
(8) vier
```

Ergebnis–PDF (Output)

Überschrift 3. Ordnung

Nummerierte Liste, geht auch so (ist aber mehr Arbeit, sollte mal ein Umsortieren erforderlich werden):

1. eins
2. zwei
3. drei

Nummerierte Liste mit Offset:

7. eins
8. zwei
9. drei
10. vier

Nummerierte Liste mit Offset:

(7) eins
(8) zwei
(9) drei
(10) vier

Markdown-Quelltext (Input)

```
Nummerierte Liste mit römischen
Ziffern:

i. eins
i. zwei
i. drei
i. vier
i. fünf
i. sechs
```

```
Einfacher *Code-Block*:

    # An jeden Zeilenbeginn 4
    # (vier) Leerzeichen. Dann
    # den restlichen Code...
```

```
Nummerierte Liste mit
Groß-Buchstaben:

A.  eins (hier bitte 2 Leerzeichen
    nach dem "A."!)
A.  zwei
A.  drei
```

```
Nummerierte Liste, noch 'n Stil:

a. eins
a. zwei
a. drei
```

Ergebnis–PDF (Output)

Nummerierte Liste mit römischen Ziffern:

i. eins
ii. zwei
iii. drei
iv. vier
v. fünf
vi. sechs

Einfacher *Code-Block*:

```
# An jeden Zeilenbeginn 4
# (vier) Leerzeichen. Dann
# den restlichen Code...
```

Nummerierte Liste mit Groß-Buchstaben:

A. eins (hier bitte 2 Leerzeichen nach dem “A.”!)
B. zwei
C. drei

Nummerierte Liste, noch ’n Stil:

a. eins
b. zwei
c. drei

Markdown-Quelltext (Input)

```
Nummerierte Liste: so geht's
ebenfalls:

a. eins
b. zwei
c. drei


...oder auch so:

(g) eins
(h) zwei
(h) drei
(h) vier


Nummerierte Liste, auch so geht's:

a) eins
b) zwei
c) drei


Nummerierte Liste, so geht's,
falls die Liste mit einem Offset
starten soll:

c. eins
d. zwei
d. drei
d. vier
```

Ergebnis–PDF (Output)

Nummerierte Liste: so geht's ebenfalls:

a. eins
b. zwei
c. drei

. . . oder auch so:

(g) eins
(h) zwei
(i) drei
(j) vier

Nummerierte Liste, auch so geht's:

a) eins
b) zwei
c) drei

Nummerierte Liste, so geht's, falls die Liste mit einem Offset starten soll:

c. eins
d. zwei
e. drei
f. vier

Markdown-Quelltext (Input)

```
Geschachtelte Liste:

* alpha
* beta
    1. eins
        A.  Erstens
        A.  Zweitens
    1. zwei
* gamma
```

Ergebnis–PDF (Output)

Geschachtelte Liste:

- alpha
- beta
    1. eins
        A. Erstens
        B. Zweitens
    2. zwei [. . . .]

Markdown-Quelltext (Input)

```
Eingerücktes Zitat:

> Die Zitat-Zeilen einfach mit `>`
> plus einem Leerzeichen einleiten.
```

Ergebnis–PDF (Output)

Eingerücktes Zitat:

> Die Zitat-Zeilen einfach mit > plus einem Leerzeichen einleiten.

Markdown-Quelltext (Input)

```
Sogenannter *fenced* Code-Block:
~~~
# Erfordert keine Leerzeichen am
# Zeilenbeginn. Einleitung und Be-
# endigung durch jeweils (mind.)
# drei "~"-Zeichen (Tilde),
# jeweils auf einer eigenen Zeile.
# Vorteile von 'fenced' Code-
# Blocks: man kann über zusätzli-
# che Attribute bewirken, dass
# pandoc die jeweilige Program-
# miersprache per Syntax-High-
# lighting übersichtlicher dar-
# stellt.
~~~
```

Ergebnis–PDF (Output)

Sogenannter *fenced* Code-Block:

```
# Erfordert keine Leerzeichen am
# Zeilenbeginn. Einleitung und Be-
# endigung durch jeweils (mind.)
# drei "~"-Zeichen (Tilde),
# jeweils auf einer eigenen Zeile.
# Vorteile von 'fenced' Code-
# Blocks: man kann über zusätzli-
# che Attribute bewirken, dass
# pandoc die jeweilige Program-
# miersprache per Syntax-High-
# lighting übersichtlicher dar-
# stellt.
```

Markdown-Quelltext (Input)

````
Fenced Code-Block mit Bash und
entsprechendem Syntax-Highlighting:
~~~ {.bash}
#!/bin/bash
# Hier ist ein Kommentar
i=8
for j={7..11}; do
   j=$(( ${i} + ${j} ))
   i=$(( ${i} + 1 ))
   echo "  j is now: ${j}
   echo "  i is now: ${i}"
done                      "
~~~
````

Ergebnis–PDF (Output)

Fenced Code-Block mit Bash und entsprechendem Syntax-Highlighting:

```bash
#!/bin/bash
# Hier ist ein Kommentar
i=8
for j={7..11}; do
   j=$(( ${i} + ${j} ))
   i=$(( ${i} + 1 ))
   echo "  j is now: ${j}"
   echo "  i is now: ${i}"
done
```

Fußnoten

Fußnoten kann man auf mehrere Arten im Markdown kodieren:

1. *Inline*-Fußnoten
2. Fußnoten mit *Kurz-Identifier*
3. Fußnoten mit *Lang-Identifier*

Wiederholung: *Inline*-Fußnoten

Hier nochmals eine Fußnote, die im Markdown-Quelltext als sogenannte *inline*[2]-Note angelegt ist).

Diese Fußnote kommt vom folgenden Markdown:

```
Hier nochmals eine Fußnote, die im Markdown-Quelltext als sogenannte
*inline*^[Fußnoten sind inline viel leichter zu schreiben, finde
ich... Denn man muß sich nicht erst einen eindeutigen *Identifier*
überlegen, diesen zweimal tippen und für's zweite Mal nach unten
```

[2] Fußnoten sind inline viel leichter zu schreiben, finde ich... Denn man muß sich nicht erst einen eindeutigen *Identifier* überlegen, diesen zweimal tippen und für's zweite Mal nach unten gehen, um dort die eigentliche Fußnote zu schreiben. Man legt diese an beliebigen Stellen des Textes einfach an, indem man das Zeichen `^` tippt, und direkt anschließend daran in eckigen Klammern (`[...]`) den Text, den die Fußnote enthalten soll.

```
gehen, um dort die eigentliche Fußnote zu schreiben. Man legt
diese an beliebigen Stellen des Textes einfach an, indem man das
Zeichen `^` tippt, und direkt anschließend daran in eckigen Klammern
(`[...]`) den Text, den die Fußnote enthalten soll.]-Note angelegt ist).
```

Vertiefung: Fußnoten mit kurzen oder langen Identifiern

Hier kommt eine weitere Fußnote[3], und dann noch eine[4], beide diesmal jedoch nicht "inline" angelegt, sondern durch sogenannte *Identifier* markiert.

Das Markdown für die letzten beiden Fußnoten sieht so aus:

```
Hier kommt eine weitere Fußnote[^k], und dann noch
eine[^langer-fussnoten_identifier], beide diesmal jedoch nicht
"inline" angelegt, sondern durch sogenannte *Identifier* markiert.

[^k]: Diese Fußnote wurde mit einem "kurzen" *Identifier* versehen.
Solch kurze Identifier bestehen aus einem einzigen alphanumerischen
Zeichen, in diesem Falle *k* (siehe das dazugehörige Markdown).

[^langer-fussnoten_identifier]: Diese Fußnote wurde mit einem langen
*Identifier* versehen. Außerdem besteht sie aus drei Absätzen.

    Lange Identifier bestehen aus einem beliebig langen Wort, das
auch "Dashes" oder "Underscores" beinhalten darf. Dass man die
(kurzen oder langen) Identifier in eckige Klammern einzufassen hat,
macht hoffentlich ein Blick auf das konkrete Markdown des Beispiels
bereits klar, oder?

    Fußnoten-Absätze muß man im Markdown-Quelltext per Leerzeile
voneinander trennen und die erste Zeile des jeweiligen Absatzes ist
um jeweils vier Leerzeichen einzurücken. Sonst würde dies im Output
nämlich kein Fußnoten-Absatz werden, sondern ein normaler Absatz im
Text...
```

[3] Diese Fußnote wurde mit einem "kurzen" *Identifier* versehen. Solch kurze Identifier bestehen aus einem einzigen alphanumerischen Zeichen, in diesem Falle *k* (siehe das dazugehörige Markdown).

[4] Diese Fußnote wurde mit einem langen *Identifier* versehen. Außerdem besteht sie aus drei Absätzen.

Lange Identifier bestehen aus einem beliebig langen Wort, das auch "Dashes" oder "Underscores" beinhalten darf. Dass man die (kurzen oder langen) Identifier in eckige Klammern einzufassen hat, macht hoffentlich ein Blick auf das konkrete Markdown des Beispiels bereits klar, oder?

Fußnoten-Absätze muß man im Markdown-Quelltext per Leerzeile voneinander trennen und die erste Zeile des jeweiligen Absatzes ist um jeweils vier Leerzeichen einzurücken. Sonst würde dies im Output nämlich kein Fußnoten-Absatz werden, sondern ein normaler Absatz im Text...

Tabellen

Der Einführungs-Artikel hatte bereits gezeigt, wie Markdown für *'Simple Tables'* aussieht.

Wiederholung: *Simple Table*

Aus folgendem Markdown. . .

```
  Right     Left     Center      Default
-------     ------ ----------    -------
   1245       1245      1245        1245
 012345     012345  012345       012345
```

. . . erzeugt `pandoc` diese Darstellung einer Tabelle:

Table 2: Simple Table

Right	Left	Center	Default
1245	1245	1245	1245
012345	012345	012345	012345

Hinsichtlich der Spalten-Ausrichtung gelten die folgenden vier Grundregeln:

1. Falls die Strich-Linie auf der rechten Seite bündig mit dem Kopf-Text ist, jedoch auf der linken Seite übersteht, dann ist die Spalte rechtsbündig.
2. Falls die Strich-Linie auf der linken Seite bündig mit dem Kopf-Text ist, jedoch auf der rechten Seite übersteht, dann ist die Spalte linksbündig.
3. Falls die Strich-Linie relativ zur Länge des Kopf-Textes auf beiden Seiten übersteht, dann ist die Spalte zentriert.
4. Falls die Strich-Linie auf beiden Seiten bündig zum Kopf-Text ist, kommt die standard-mäßig voreingestellte Spalten-Ausrichtung zu Tragen (meistens wird dies Linkbündigkeit sein).

Dabei ist es nicht unbedingt erforderlich, den Inhalt der Tabellen-Zellen ebenfalls exakt auszurichten (wenn man's macht, sieht der Quelltext halt schöner aus. . .)

Drittes Tabellen-Beispiel

(Das erste Tabellen-Beispiel in diesem Artikel war übrigens die sich von S. 1–6 erstreckende tabellarische Übersicht. . .)

```
                             Einheit   **Vulcain I**   **Vulcain II**
-----------------------------  --------  --------------  ----------------
Spezifischer Impuls im Vakuum  [*s*]          431,2          433,1
                     LH~2~/LO~2~  [*-*]           5,34            6,1
              Gesamtmassenstrom  [*kg/s*]         271            319
               Brennkammerdruck  [*bar*]          110            116
  Düsenquerschnittsverhältnis  [*-*]           45             60

Table: *Die dritte Tabelle.*  (Tabellen-Unterschrift ist teilweise\
*kursiv*.)
```

Table 3: *Die dritte Tabelle.* (Tabellen-Unterschrift ist teilweise *kursiv*.)

	Einheit	**Vulcain I**	**Vulcain II**
Spezifischer Impuls im Vakuum	[*s*]	***431,2***	433,1
LH_2/LO_2	[-]	***5,34***	6,1
Gesamtmassenstrom	[*kg/s*]	***271***	319
Brennkammerdruck	[*bar*]	***110***	116
Düsenquerschnittsverhältnis	[-]	***45***	60

Änderungen: (1) Keine Überschrift für die linke Spalte; beide rechte Spalten sind zentriert. – (2) Die *Einheiten*- und *Vulcain I/II*-Spalten haben zusätzliche Formatierungen.

Vierte Tabelle

```
-----------------------------  ---------  -----------  ------------
Spezifischer Impuls im Vakuum  [*s*]           431,2          433,1
                     LH~2~/LO~2~  [*-*]            5,34            6,1
              Gesamtmassenstrom  [*kg/s*]          271            319
               Brennkammerdruck  [*bar*]           110            116
  Düsenquerschnittsverhältnis  [*-*]            45             60
-----------------------------  ---------  -----------  ------------

Table: Die vierte Tabelle (ohne Tabellen-Kopf: dann sind unten, als\
Tabellen-Abschluss, ebenfalls Linien im Markdown erforderlich!)
```

Table 4: Die vierte Tabelle (ohne Tabellen-Kopf: dann sind unten, als Tabellen-Abschluss, ebenfalls Linien im Markdown erforderlich!)

Spezifischer Impuls im Vakuum	[*s*]	431,2	433,1

LH_2/LO_2	[-]	5,34	6,1
Gesamtmassenstrom	[kg/s]	271	319
Brennkammerdruck	[bar]	110	116
Düsenquerschnittsverhältnis	[-]	45	60

Änderungen: Gar keine Spaltenüberschriften – in diesem Falle ist die zweite, jeweils die Spalten markierende (korrekt unterbrochene) und die Tabelle abschließende Linie erforderlich. Sonst erkennt der Markdown-Parser nicht, dass da eine Tabelle sein soll.

Fünfte Tabelle

```
                             Einheit    **Vulcain I**   **Vulcain II**
----------------------------  --------  -------------  --------------
                      Vakuumschub  [*kN*]     1145           1350
Spezifischer Impuls im Vakuum  [*s*]      431,2          433,1
                    LH~2~/LO~2~  [*-*]      5,34           6,1
              Gesamtmassenstrom  [*kg/s*]   271            319
               Brennkammerdruck  [*bar*]    110            116
  Düsenquerschnittsverhältnis  [*-*]      45             60
----------------------------  --------  -------------  --------------

Table: Fünfte Tabelle (mit Spalten-Überschriften: dann sind Linien unten\
im Markdown optional.)
```

Table 5: Fünfte Tabelle (mit Spalten-Überschriften: dann sind Linien unten
im Markdown optional.)

	Einheit	**Vulcain I**	**Vulcain II**
Vakuumschub	[kN]	1145	1350
Spezifischer Impuls im Vakuum	[s]	431,2	433,1
LH_2/LO_2	[-]	5,34	6,1
Gesamtmassenstrom	[kg/s]	271	319
Brennkammerdruck	[bar]	110	116
Düsenquerschnittsverhältnis	[-]	45	60

Änderungen: Alle Spalten (außer der linken) sind linksbündig.

Sechste Tabelle

```
                              Einheit   **Vulcain I**   **Vulcain II**
----------------------------- --------- --------------- ----------------
                     Vakuumschub  [*kN*]            1145             1350
Spezifischer Impuls im Vakuum  [*s*]             431,2            433,1
                   LH~2~/LO~2~  [*-*]              5,34              6,1
              Gesamtmassenstrom  [*kg/s*]           271              319
               Brennkammerdruck  [*bar*]            110              116
  Düsenquerschnittsverhältnis  [*-*]              45               60
----------------------------- --------- --------------- ----------------

Table: **Die sechste Tabelle (ebenfalls ohne Tabellenkopf -- deshalb\
sind unten jetzt ebenfalls Linien erforderlich!)**
```

Table 6: **Die sechste Tabelle (ebenfalls ohne Überschrift – deshalb unten ebenfalls Linien!)** Die Tabellen-Unterschrift ist teilweise **fett**...

	Einheit	**Vulcain I**	**Vulcain II**
Vakuumschub	[kN]	1145	1350
Spezifischer Impuls im Vakuum	[s]	431,2	433,1
LH_2/LO_2	[-]	5,34	6,1
Gesamtmassenstrom	[kg/s]	271	319
Brennkammerdruck	[bar]	110	116
Düsenquerschnittsverhältnis	[-]	45	60

Änderungen: Alle Spalten sind rechtsbündig.

Siebente Tabelle (Extension): `multiline_tables`, `table_captions`

Multiline tables ermöglichen es, sowohl Tabellenzeilen und Spaltenköpfe als auch Tabellen-Unterschriften über mehrere Zeilen laufen zu lassen. Zudem lassen sich die einzelnen Spalten-Breiten innerhalb gewisser Grenzen kontrollieren. – Hier ein erstes Beispiel:

```
-------------------------------------------------------------------------
Zentrierter  Ausgerichtet                    Rechts Links
Spaltenkopf  ohne besondere               ausgerichtet ausgerichtet
             Vorgabe (ergo
             Default)
------------ -------------- ------------------------- -----------------
```

```
Erste        Zeile                                12.0 Beispiel für eine
                                                       Zelle, die sich
                                                       über mehrere
                                                       Zeilen erstreckt.

Zweite       Zeile                                 5.0 Hier noch eine.
                                                       Man beachte
                                                       unbedingt den
                                                       Leerraum im
                                                       Markdown
                                                       zwischen den
                                                       Tabellenzeilen!
-------------------------------------------------------------------------

Table: Tabellen-Unterschrift.
Auch
diese
darf sich im Markdown über mehrere Zeilen erstrecken.
```

Table 7: Tabellen-Unterschrift. Auch diese darf sich im Markdown über mehrere Zeilen erstrecken.

Zentrierter Spaltenkopf	Ausgerichtet ohne besondere Vorgabe (ergo Default)	Rechts ausgerichtet	Links ausgerichtet
Erste	Zeile	12.0	Beispiel für eine Zelle, die sich über mehrere Zeilen erstreckt.
Zweite	Zeile	5.0	Hier noch eine. Man beachte unbedingt den Leerraum im Markdown zwischen den Tabellenzeilen!

Hinweis: Man beachte, wie die im Markdown vorhandenen Zeilen-Umbrüche der Spalten-Köpfe, Tabellen-Zellen und Tabellen-Untertitel sich beim Rendern nach PDF geändert haben.

Multiline-Tables funktionieren im Prinzip ebenso wie Simple Tables, mit folgen-

den Unterschieden in den Feinheiten:

- Sie müssen mit einer Zeile von *'Dashes'* anfangen (also die oben gezeigten gestrichelten Linien, die wie bei den Simple Tables an den Spalten-Grenzen unterbrochen ist).
- Erst danach kommt der Tabellenkopf-Text (außer, dieser soll ganz entfallen).
- Sie müssen mit einer Zeile von *'Dashes'* abschließen (Strichlinie), gefolgt von einer Leerzeile.
- Die Tabellenzeilen muss man durch Leerzeilen voneinander trennen.
- Nach dem Rendern nehmen die Tabellen immer die gesamte verfügbare Seitenbreite ein (nicht nur den gerade erforderlichen Platz, wie die Simple Tables).
- Tabellen-Zellen, die sich über mehrere Spalten oder mehrere Zeilen erstrecken, sind allerdings nicht unterstützt.

  ACHTUNG: Falls man eine der Tabellen verwenden möchte, die mit einer oder mehreren *Extension*-Anmerkung gekennzeichnet sind, dann kann man beim Übersetzen nicht mehr das einfache `-f markdown` bzw. `--from=markdown` verwenden. Man muss `pandoc` dann über diese Erweiterung informieren, indem man `-f markdown+ext_1+ext_2` bzw. `--from=markdown+ext_1+ext_2` spezifiziert.

  Im konkreten Fall also: beim Konvertieren `--f markdown+multiline_tables` verwenden!

Achte Tabelle

Spaltenköpfe kann man sowohl bei Multiline-Tables als auch bei Simple Tables weglassen:

```
------------ -------------- -------------- ----------------------------
Erste         Zeile                    12.0 Beispiel für eine Zelle,
                                            die sich über mehrere
                                            Zeilen erstreckt.

Zweite        Zeile                     5.0 Hier ist noch eine. Man
                                            beachte unbedingt den
                                            Leerraum im Markdown
                                            zwischen den Tabellenzeilen!
------------ -------------- -------------- ----------------------------

: Hier eine Multiline-Table ohne Spaltenköpfe. Außerdem befindet sich\
hier kein *"Table:"*-Schlüsselwort zu Beginn der Tabellen-Bezeichnung\
im Quelltext, sondern lediglich ein Doppelpunkt. -- Fällt auf, was das\
an der Ausgabe ändert?? (*Hint:* Gar nichts!)
```

Das ergibt folgende Tabelle:

Table 8: Hier eine Multiline-Table ohne Spaltenköpfe. Außerdem befindet sich
hier kein *"Table:"*-Schlüsselwort zu Beginn der Tabellen-Bezeichnung
im Quelltext, sondern lediglich ein Doppelpunkt. – Fällt auf, was das
an der Ausgabe ändert?? (*Hint:* Gar nichts!)

Erste	Zeile	12.0	Beispiel für eine Zelle, die sich über mehrere Zeilen erstreckt.
Zweite	Zeile	5.0	Hier ist noch eine. Man beachte unbedingt den Leerraum im Markdown zwischen den Tabellenzeilen!

Es ist möglich, dass eine Multiline-Table nur eine einzige Zeile enthält. In diesem Fall sollte jener Tabellenzeile im Markdown ebenfalls eine Leerzeile folgen (und zuletzt eine Zeile mit 'Dashes', um die Tabelle abzuschließen). Ansonsten würde `pandoc` diese Zeile als Simple Table interpretieren.

WICHTIG: Bei Multiline-Tables versucht `pandoc`, die jeweiligen Spaltenbreiten zu erkennen; die `pandoc`-Ausgabemodule versuchen, diese Breiten relativ zueinander wiederzugeben. Wer also beim Anblick des Output eine Spalte als zu schmal empfindet, kann die dadurch korrigieren, dass er/sie die Spalte im Markdown-Quellcode breiter macht. – Umgesetzt auf das letzte Beispiel, kann das in etwa so aussehen:

Multiline-Table mit geänderten Spalten-Breiten

```
------- ------ ----- ---------------------------------------------------
Erste   Zeile   12.0 Beispiel für eine Zelle,
                     die sich über mehrere
                     Zeilen erstreckt.

Zweite  Zeile        Hier ist noch eine. Man
                 5.0 beachte unbedingt den
                     Leerraum im Markdown
                     zwischen den Tabellenzeilen!
------- ------ ----- ---------------------------------------------------
```

```
Table: diesmal mit geänderten Spaltenbreiten! (Relative Spaltenbreiten
werden durch die jeweiligen Längenverhältnisse der gestrichelten
Teil-Linien festgelegt, nicht durch die Breiten der Zellen-Texte.)
Zelleninhalte werden nach oben ausgerichtet. **ACHTUNG:** Man sollte
Zelleninhalte trotzdem immer in der obersten Zeile beginnen lassen.
Ansonsten könnten sie unter bestimmten Bedingungen ganz verschwinden.
```

Table 9: diesmal mit geänderten Spaltenbreiten! (Relative Spaltenbreiten werden durch die jeweiligen Längenverhältnisse der gestrichelten Teil-Linien festgelegt, nicht durch die Breiten der Zellen-Texte.) Zelleninhalte werden nach oben ausgerichtet. **ACHTUNG:** Man sollte Zelleninhalte trotzdem immer in der obersten Zeile beginnen lassen. Ansonsten könnten sie unter bestimmten Bedingungen ganz verschwinden.

Erste	Zeile	12.0	Beispiel für eine Zelle, die sich über mehrere Zeilen erstreckt.
Zweite	Zeile	5.0	Hier ist noch eine. Man beachte unbedingt den Leerraum im Markdown zwischen den Tabellenzeilen!

Dann noch: *Grid Tables*

Extension: `grid_tables`, `table_caption`

Bei den Grid-Tables trennt eine Zeile mit Gleichheitszeichen ('=') den Tabellenkopf vom -körper. Diese Zeile kann man bei Tabellen ohne Kopf weglassen. Vorteile und Besonderheiten von Grid Tables:

- Die Zellen der Grid Tables dürfen beliebige Block-Elemente enthalten (also mehrere Absätze, oder Code-Blocks, Listen usw.
- Rechtsbündige oder zentrierte Ausrichtung von Spalten ist allerdings nicht unterstützt. Grid Tables sind immer linksbündig.
- Man kann die Spalten-Breiten von Grid Tables (sowie ihre Gesamt-Breite) beeinflussen, indem man die Zellen breiter oder schmaler macht.
- Wer mit dem `emacs`-Editor umgehen kann, kann solche Tabellen mit Gitter im sog. *table mode* sehr einfach erstellen.

Sie sehen wie folgt aus.

Schmale Grid Table

Zuerst eine schmale Variante:

```
: Beispiel für eine schmale *Grid Table* (Gitter-Tabelle)

+----------+---------+----------------------------------+
|**Frucht**|**Preis**|**Vorteile**                      |
+==========+=========+==================================+
|Bananen   |€1.34    |- Besonders toll wegen:           |
|          |         |- integriertem Verpackungsmaterial|
|          |         |- heller Farbe                    |
+----------+---------+----------------------------------+
|Orangen   |€2.10    |Beliebt, weil:                    |
|          |         |- heilt Skorbut                   |
|          |         |- schmeckt gut                    |
+----------+---------+----------------------------------+
```

Table 10: Beispiel für eine schmale *Grid Table* (Gitter-Tabelle)

Frucht	**Preis**	**Vorteile**
Bananen	€1.34	• Besonders toll wegen: • integriertem Verpackungsmaterial • heller Farbe
Orangen	€2.10	Beliebt, weil: - heilt Skorbut - schmeckt gut

Breite Grid Table

Dieselbe Gitter-Tabelle, im Markdown etwas breiter angelegt…

```
: Beispiel für eine breite *Grid Table* (Gitter-Tabelle)

+------------+-------------+-----------------------------------------+
| **Frucht** | **Preis**   | **Vorteile**                            |
+============+=============+=========================================+
| Bananen    | €1.34       | Besonders toll wegen:                   |
|            |             |                                         |
|            |             | - integriertem Verpackungsmaterial      |
|            |             | - heller Farbe                          |
+------------+-------------+-----------------------------------------+
| Orangen    | €2.10       | Beliebt, weil:                          |
|            |             |                                         |
```

```
|            |              | - heilt Skorbut                          |
|            |              | - schmeckt gut                           |
+------------+--------------+------------------------------------------+
```

. . . kommt auch auf der PDF-Seite breiter raus:

Table 11: Beispiel für eine breite *Grid Table* (Gitter-Tabelle)

Frucht	**Preis**	**Vorteile**
Bananen	€1.34	Besonders toll wegen: • integriertem Verpackungsmaterial • heller Farbe
Orangen	€2.10	Beliebt, weil: • heilt Skorbut • schmeckt gut

Zu guter Letzt: *Pipe tables*

Extension: `pipe_tables`, `table_captions`

Pipe tables bereitet man so vor:

```
| Rechts | Links| Default | Center |
|-------:|:-----|---------|:------:|
|   12   |  12  |    12   |    12  |
|  123   |  123 |   123   |   123  |
|    1   |    1 |     1   |     1  |

Table: Beispiel der *Pipe Table*-Syntax.
```

Table 12: Beispiel der *Pipe Table*-Syntax.

Rechts	Links	Default	Center
12	12	12	12
123	123	123	123
1	1	1	1

Diese Syntax ist dieselbe wie im Markdown-Dialekt *PHP markdown*. Die Pipe-Zeichen sind zu Anfang und Ende der Zeilen optional, aber *zwischen* den Spalten sind sie als Trennzeichen unbedingt erforderlich. Besonderheiten der Pipe Tables:

- Die Positionen der Doppelpunkte legen die jeweilige Spalten-Ausrichtung fest.
- Den Tabellenkopf kann man weglassen, aber die Linie mit den Doppelpunkten muss bleiben, da sie die Spalten-Ausrichtung definiert.

Da die Pipe-Zeichen die Spaltengrenzen definieren, müssen diese nicht unbedingt vertikal ausgerichtet sein wie im obigen Beispiel (es sieht halt im Quelltext schöner aus!). Folglich ist es völlig 'legal' im Sinne dieser Syntax die folgende 'hässliche' Tabellen zu schreiben...

'Hässliche' Pipe-Tabelle

```
Frucht|Preis
-----|-----:
Apfel|2.05
Pfirsich|1.37
Orange|3.09

Table: diesmal mit 'hässlicher' Syntax im Markdown-Code...
```

Table 13: diesmal mit 'hässlicher' Syntax im Markdown-Code...

Frucht	Preis
Apfel	2.05
Pfirsich	1.37
Orange	3.09

Weitere Besonderheiten von Pipe Tables:

- Man muss sich nicht besonders anstrengen, sie zu schreiben, da die 'hässliche' Form viel Schreib- und Editier-Faulheit toleriert.
- Die Zellen von Pipe-Tables können keine Block-Elemente (wie Absätze und Listen) enthalten, und sie können auch nicht mehrere Zeilen enthalten.

Anmerkung: Pandoc kann auch mit Pipe-Tables der folgenden Form umgehen:

```
|EINS |ZWEI   |
|-----+-------|
```

```
|Meine|Tabelle|
|ist  |schön  |
```

EINS	ZWEI
Meine	Tabelle
ist	schön

Besonderheiten: * Der Unterschied zum vorigen Beispiel liegt darin, dass hier ein `+`-Zeichen zur Anwendung kommt anstelle eines `|`-Zeichens. * Jetzt kann man, um eine von der Voreinstellung abweichende Spaltenausrichtung zu erhalten, ebenfalls Doppelpunkte wie zuvor gezeigt verwenden.

Ein letzter Hinweis: wie man an obigen Beispiel sieht: man kann Tabellen-Unterschriften auch *vor* der Tabelle einfügen statt danach. Dann stehen sie im Ergebnis trotzdem nicht ober-, sondern immer noch unterhalb der Tabelle. Das gilt für alle Tabellen-Arten.

CSV nach Tabellen: mit `--filter=pandoc-csv2table` (NEU!)

Seit einigen Monaten erhältlich: ein externer Filter, der aus CSV (*Comma Separated Values*) Tabellen-Code erzeugt. Er lässt sich auf der Pandoc-Kommandozeile per Aufruf `--filter=pandoc-csv2table` einbinden.

Geschrieben hat den Filter Wasif Hasan Baig. Der Quellcode ist auf GitHub: https://github.com/baig/pandoc-csv2table.

Installieren lässt er sich auf zweierlei Arten:

1. Per `cabal install` (Haskell Paket-Management).
2. Manuell, per Download und Hinterlegen im “Pfad”.

Nutzen

Insbesondere bei größeren Tabellen erspart dieser Filter die mühsame Arbeit des manuellen Schreibens von Grid-, Pipe- oder Multiline-Tabellen.

Das lohnt sich schon ab einer Tabellen-Größe von 3-4 Zeilen mit 3-4 Spalten.

Insbesondere effektiv ist seine Verwendung, falls die Tabellen-Daten ohnehin bereits in einem Excel- oder LibreOffice Calc-Dokument vorliegen und dort erst aufwändig herauszupflücken wären.

Denn der Filter kann selbstverständlich auch aus den CSV-Daten wieder schön formatierte Markdown-Grid-, Pipe- oder Multiline-Tabellen herstellen.

Installation per `cabal-install`

Voraussetzung hierfür ist, dass Pandoc selbst bereits per Cabal installiert ist und in einer aktuellen Version vorliegt. Dann besteht dieser Weg einfach aus folgendem Befehl:

```
cabal update
cabal install pandoc-csv2table
```

Manuelle Installation

Voraussetzung ist eine einigermaßen aktuelle Pandoc-Version.

Dann kann man die folgende Datei herunterladen:

- https://gist.github.com/baig/b69e3146251bd90d12e7

Datei ausführbar machen per `chmod a+x pandoc-csv2table.hs` und im `${PATH}` hinterlegen, z.B. in `${HOME}/bin/`.

Verwendung

Der Filter "misbraucht" die bekannte Markdown-Syntax für `fenced_code_blocks`. Anstatt jedoch tatsächlich eine Code Block innerhalb der Markierungen zu erwarten, erwartet er dort Zeilen, die CSV-Felder darstellen. Innerhalb der CSV-Felder kann man die Texte sogar mit Markdown "pimpen", um in der Tabelle die entsprechenden Formatierungen zu erreichen!

Normale Code Blocks innerhalb des Markdowns lässt er in Ruhe. Er springt erst an, wenn die einleitende "Fence"-Markierung das Attribut `{.table}` aufweist.

Einfaches Beispiel

Das Markdown könnte so aussehen:

```
**Frucht**,**Preis**
Apfel,2.05
Pfirsich,1.37
Orange,3.09
```

Der Aufruf `pandoc --filter=pandoc-csv2table.hs` (bei manueller Installation des Filters) bzw. `pandoc --filter=pandoc-csv2table` (nach Installation via Cabal) macht daraus dann eine Tabelle mit der Default-Formattierung Pandocs.

Komplexeres Beispiel

Ausser des unbedingt erforderlichen Attributs `{.table}` kann man dem "fenced table block" weitere optionale Attribute verpassen. Und die machen dann die wirklich interessanten Features dieses Filters aus! Es sind dies:

1. **`caption="<text>"`** :
   Dies setzt eine Tabellen-Unter- oder Überschrift.

2. **`type="grid"`** :
   Dies weist den Filter an, eine `grid_table` zu erzeugen. Weitere zulässige Werte sind: `type="multiline"`, `type="simple"` und `type="pipe"`

3. **`header="yes"`** :
   Anweisung an den Filter, die Felder der ersten Zeile als Spalten-Köpfe zu behandeln. Bei `"no"` erzeugt man eine Tabelle on Spalten-Überschriften.

4. **`aligns=RCLRD`** :
   Anweisung an den Filter, die erste Spalte rechtsbündig auszurichten (`R`). Die zweite Spalte soll zentriert sein (`C`), die dritte linksbündig (`L`), die vierte wieder rechtsbündig (`R`) und die letzte soll die Default-Ausrichtung erhalten (`D`).

5. **`source="/pfad/zu/my.csv"`** :
   Bei Verwendung dieses Attributs kann der "fenced table block" sogar leer sein. Der Filter liest die CSV-Daten dann aus der bezeichneten Datei ein.

Im einfachste Fall könnte man also folgendes Markdown schreiben. Es verwendet einen leeren table block und referenziert die Datei `my.csv`:

Dann kann man folgendes Markdown schreiben:

```
~~~ {.table caption="Die sollte eine `grid_table` (Gitter-Tabelle) ergeben" type="grid" alig
~~~
```

Hier derselbe Tabellen-Block, jedoch ohne Referenzierung auf eine externe CSV-Datei:

```
~~~ {.table caption="Dies sollte eine `grid_table` (Gitter-Tabelle) ergeben" type="grid" ali
**Fruchtsorte(R)**, *Menge(C)*, ***Preis(R)***, **`Herkunft`(L)**, `Qualität`(C), verpackt(
Äpfel, 15,"3,24", Spanien, exzellent, ***ja***, ja
Orangen, 12,"2,22", Deutschland, **sauer**, nein, bald
~~~
```

Jetzt die Nagelprobe: Aufruf des Pandoc-Kommandos.

```
pandoc --filter=pandoc-csv2table --from=markdown --to=markdown \
       --output=gittertabelle.md myinput.md
```

Ergebnis:

```
+-----------+--------+----------+-----------+----------+---------+-----------------+
| **Fruchts | *Menge | ***Preis | **`Herkun | `Qualitä | verpack | ausverkauft?(C) |
| orte(R)** | (C)*   | (R)***   | ft`(L)**  | t`(C)    | t(R)    |                 |
+===========+========+==========+===========+==========+=========+=================+
| Äpfel     | 15     | 3,24     | Spanien   | exzellen | ***ja** | ja              |
|           |        |          |           | t        | *       |                 |
+-----------+--------+----------+-----------+----------+---------+-----------------+
| Orangen   | 12     | 2,22     | Deutschla | **sauer* | nein    | bald            |
|           |        |          | nd        | *        |         |                 |
+-----------+--------+----------+-----------+----------+---------+-----------------+

: Dies sollte eine `grid_table` (Gitter-Tabelle) ergeben
```

Das sieht (noch) nicht so schön aus, diese Gitter-Tabelle "funktioniert" nicht: denn die Umbrüche innerhalb der Worte würden im Ausgabe-Dokument zu Wort-Lücken führen.

Um diese unerwünschten Zeilen-Umbrüche innerhalb der Gitterzellen zu eliminieren, fügt man dem Pandoc-Kommando einfach den Parameter `--columns=112` hinzu. Dann verwendet Pandoc (mindestens) eine Zeilenbreite von 112 Zeichen, um den Markdown-Text auszugeben.

Bugs

Der Filter funktioniert im Prinzip sehr gut. Die aktuelle Version (22. August 2015) hat einen Bug, der verhindert, dass er...

- ...korrekte Pipe-Tables erzeugt oder dass er
- ... bei Multiline-Tables eine gewünschte Links-Bündigkeit von Spalten erzeugt.

Der Bearbeitung-Fortschritt des entsprechenden Bug-Report lässt sich hier verfolgen:

- https://github.com/baig/pandoc-csv2table/issues/11

Mathe-Formeln à la LaTeX

Der Einführungs-Artikel hatte bereits gezeigt, dass `pandoc` schön gesetzte LaTeX-Formeln ins PDF-Ausgabeformat übertragen kann, sofern man die Formel bereits im Markdown als LaTeX-Code einbettet:

```
$\cos (2\theta) = \cos^2 \theta - \sin^2 \theta$

$\lim_{x \to \infty} \exp(-x) = 0$

$\frac{n!}{k!(n-k)!} = \binom{n}{k}$
```

So sieht dann das Ergebnis im PDF aus:

$\cos(2\theta) = \cos^2 \theta - \sin^2 \theta$

$\lim_{x \to \infty} \exp(-x) = 0$

$\frac{n!}{k!(n-k)!} = \binom{n}{k}$

Vertiefung: Formeln zentrieren

Wer die Formeln auf ihren jeweiligen Zeilen zentrieren möchte, nimmt statt einfachen Dollarzeichen doppelte – `$$`:

```
$$\forall x \in X, \quad \exists y \leq \epsilon$$

$$\cos (2\theta) = \cos^2 \theta - \sin^2 \theta$$

$$\lim_{x \to \infty} \exp(-x) = 0$$

$$\frac{n!}{k!(n-k)!} = \binom{n}{k}$$

$$\frac{\frac{1}{x}+\frac{1}{y}}{y-z}$$

$$\int_0^\infty \mathrm{e}^{-x}\,\mathrm{d}x$$
```

Das Ergebnis bei zentrierten Formeln kommt so raus:

$$\forall x \in X, \quad \exists y \leq \epsilon$$

$$\cos(2\theta) = \cos^2 \theta - \sin^2 \theta$$

$$\lim_{x \to \infty} \exp(-x) = 0$$

$$\frac{n!}{k!(n-k)!} = \binom{n}{k}$$

$$\frac{\frac{1}{x} + \frac{1}{y}}{y - z}$$

$$\int_0^\infty \mathrm{e}^{-x}\,\mathrm{d}x$$

Im HTML-, ODT- und DOCX-Output kommen diese Formeln allerdings nur teilweise korrekt rüber – im LaTeX-Output erscheinen sie tadellos (und ebenfalls im PDF, falls es via LaTeX erzeugt wurde).

Vertiefung: Formeln zentrieren und nummerieren

Sollte man `pandoc` dazu bringen wollen, die Formeln (für die LaTeX- oder PDF-Ausgabe) automatisch durchzunummerieren, muss das Markdown wie folgt lauten:

```
\begin{align}\cos (2\theta) = \cos^2 \theta - \sin^2 \theta\end{align}

\begin{align}\lim_{x \to \infty} \exp(-x) = 0\end{align}

\begin{align}\frac{n!}{k!(n-k)!} = \binom{n}{k}\end{align}
```

Und so sieht dann das fertige Ergebnis aus:

$$\cos(2\theta) = \cos^2 \theta - \sin^2 \theta \tag{1}$$

$$\lim_{x \to \infty} \exp(-x) = 0 \tag{2}$$

$$\frac{n!}{k!(n-k)!} = \binom{n}{k} \tag{3}$$

Wie man sieht, kommt hier eine `"align"`-Umgebung aus der LaTeX-Welt zum Einsatz. Dies funktioniert, da hier ein Mini-Schnipsel Original-LaTeX-Codes innerhalb des Markdowns die Pandoc-Erweiterung zum Einbinden von **`+raw_tex`** automatisch aktiviert.

Die **`+raw_tex`**-Erweiterung ist allerdings funktionslos für alle anderen Ausgabe-Formate! Wer dieses Dokument in einem anderen Format liest, sieht daher oben keine Nummerierung der Formeln – und je nach Format: überhaupt keine Formel...

Umgang mit Bibliographien

Auch Literatur-Verweise lassen sich automatisiert einfügen, wenn man zuvor die richtigen Vorbereitungen getroffen hat. `pandoc` kann nicht nur Formeln, Tabellen und Bilder automatisch durchnummerieren, sondern auch aus beigefügten Referenz-Kürzeln die entsprechenden Komplett-Referenzen aus einer Bibliographie-Datei holen und diese automatisch in das Ergebnis-Dokument einbinden. Hierfür muss man lediglich Kurz-Verweise in Form von `@reference` an den entsprechenden Stellen des Markdown-Textes einfügen.[5]

Funkionierendes Markdown könnte dann wie folgt lauten:

```
\begin{align}\cos (2\theta) = \cos^2 \theta - \sin^2 \theta\end{align}

Diese Gleichung geht zurück auf Thales von Milet [@ref1, der ja bekanntermaßen auch\
die Venus von Milo [@ref1] in Stein gemeisselt hat...

\begin{align}\lim_{x \to \infty} \exp(-x) = 0\end{align}

Diese Gleichung geht zurück auf Oberstudienrat Oster [@ref2].

\begin{align}\frac{n!}{k!(n-k)!} = \binom{n}{k}\end{align}

Diese Gleichung geht zurück auf eine Gemeinschaftsleistung von Elvis Presley, Madonna\
und Pink Floyd [@ref3].
Allerdings bestreitet Jane Doz [@ref4] dieses Narrativ vehement!
```

Das fertige Ergebnis:

$$\cos(2\theta) = \cos^2\theta - \sin^2\theta \tag{4}$$

[5] Dies funktioniert nur dann, wenn man eine entsprechende CSL-Datei verwendet. Ich habe zwischenzeitlich auf die IEEE-CSL umgestellt – diese stellt innerhalb des Textes nur numerische Referenzen dar.

Diese Gleichung geht zurück auf Thales von Milet [[1], der ja bekanntermaßen auch die Venus von Milo [1] in Stein gemeisselt hat...

$$\lim_{x \to \infty} \exp(-x) = 0 \tag{5}$$

Diese Gleichung geht zurück auf Oberstudienrat Oster [2].

$$\frac{n!}{k!(n-k)!} = \binom{n}{k} \tag{6}$$

Diese Gleichung geht zurück auf eine Gemeinschaftsleistung von Elvis Presley, Madonna und Pink Floyd [3]. Allerdings bestreitet Jane Doz [4] dieses Narrativ vehement!

Wo genau kommen im vorliegenden Fall die Verweise `@ref1`, `@ref2`, `@ref3` und `@ref4` eigentlich her? Dies erläutert der Abschnitt *"Vertiefung: Literaturliste automatisch erzeugen und einbinden"*, in einem späteren Abschnitt, etwas genauer...

Pandoc als HTML-nach-XYZ-Converter für das Web

Ok, die Überschrift hier übertreibt vielleicht ein wenig...

Aber falls die angepeilte Webseite nicht unbedingt Web-3.0-mäßig aufgebaut ist und viel JavaScript verwendet, dann kann man `pandoc` auch schon mal verwenden, um aus dem Web direkt HTML-Seiten nach PDF zu konvertieren.

HTML nach PDF

Folgendes Kommando sollte eigentlich gut funktionieren:

```
pandoc  -f html -o gnu-make-manual.pdf \
         http://www.gnu.org/software/make/manual/make.html
```

Mit etwas anspruchsvolleren Optionen wird daraus:

```
pandoc                          \
  --smart                       \
  --toc                         \
```

```
  --highlight-style=espresso      \
  --standalone                    \
  --self-contained                \
  -f html                         \
  -o gnu-make-manual.pdf          \
  -V geometry:paperwidth=29.7cm   \
  -V geometry:paperheight=21.0cm  \
  -V geometry:margin=0.4cm        \
   http://www.gnu.org/software/make/manual/make.html
```

Bei mir benötigte dieses Kommando ca. 9.5 Sekunden, um ein 205 Seiten umfassendes PDF im Querformat A4 (mit 4 mm Rändern rundum) zu erzeugen. Da `wget` ca. 45 Sekunden benötigte, um dieselbe HTML-Seite 10x zu holen, verbleiben für die HTML=>PDF Konvertierung noch ca. 5 Sekunden. `pandoc`'s Konvertierungsgeschwindigkeit betrug also ca. 41 Seiten pro Sekunde... nicht schlecht, oder?

HTML nach LaTeX

Man könnte nach Abwandlung des vorigen Kommandos `pandoc` auch eine Konvertierung der GNU-Make-Seite nach LaTeX versuchen lassen:

```
pandoc                            \
  --smart                         \
  --toc                           \
  --highlight-style=espresso      \
  --standalone                    \
  --self-contained                \
  --from=html                     \
  --to=latex                      \
  --output=gnu-make-manual.tex    \
  -V geometry:paperwidth=21.0cm   \
  -V geometry:paperheight=29.7cm  \
  -V geometry:margin=1.4cm        \
   http://www.gnu.org/software/make/manual/make.html
```

Wie "gut" das aus HTML erzeugte LaTeX geworden ist, kann ich leider nicht selbst beurteilen.

HTML nach ASCIIdoc

```
pandoc                            \
  --smart                         \
```

```
  --toc                           \
  --highlight-style=espresso      \
  --standalone                    \
  --self-contained                \
  -f html                         \
  -t asciidoc                     \
  -o gnu-make-manual.asciidoc     \
  -V geometry:paperwidth=21.0cm   \
  -V geometry:paperheight=29.7cm  \
  -V geometry:margin=1.4cm        \
   http://www.gnu.org/software/make/manual/make.html
```

Hier machen jetzt bestimmte Kommandozeilen-Parameter, die einfach aus dem ersten Beispiel kopiert sind, keinen Sinn mehr. Papier-Dimensionen und Ränder (und auch `--self-contained`) ergeben für ASCIIdoc-Ausgabe keinen Mehrwert. Aber mal sehen, ob's schadet... ob `pandoc` sich darüber mokiert? Ob es zu einer Fehlermeldung kommt? Ob `pandoc` damit überhaupt eine Ausgabe erzeugen kann?

Nur ein Praxistest kann das beantworten...

HTML nach ODT

```
pandoc                            \
  --smart                         \
  --toc                           \
  --highlight-style=espresso      \
  --standalone                    \
  --self-contained                \
  -f html                         \
  -t odt                          \
  -o gnu-make-manual.odt          \
  -V geometry:paperwidth=21.0cm   \
  -V geometry:paperheight=29.7cm  \
  -V geometry:margin=1.4cm        \
   http://www.gnu.org/software/make/manual/make.html
```

HTML nach Manpage

```
pandoc                            \
  --smart                         \
  --toc                           \
  --highlight-style=espresso      \
  --standalone                    \
```

```
  --self-contained               \
  -f html                        \
  -t man                         \
  -o gnu-make-manual.man         \
  -V geometry:paperwidth=21.0cm  \
  -V geometry:paperheight=29.7cm \
  -V geometry:margin=1.4cm       \
   http://www.gnu.org/software/make/manual/make.html
```

Hier gilt das oben Gesagte ebenfalls: manche der Parameter ergeben bei man-pages keinen Sinn.

Interaktiv mit `pandoc`

`pandoc` kann nicht nur mit Dateien umgehen, sondern bewältigt Ein- und Ausgaben auch per `<stdout>`/`<stdin>`. Daher eignet sich das Tool ebenso als *Filter* im schönsten Unix-Sinne, und man kann es auf interaktive Weise nutzen.

Einstieg mit LaTeX

Somit bietet es sich mir als LaTeX-Neuling ja direkt mal an, folgendes zu erforschen: *Wie kodiert man in LaTeX eine verschachtelte Liste?* Da ich weiß, wie man das in Markdown macht, tippe ich jetzt folgendes in einem Terminal ein:

```
$> pandoc -f markdown -t latex
* Eins
* Zwei
* Drei
    1. alpha
    2. beta
    1. gamma
* Vier
    1. erstens
    1. zweitens
        i. Vorname
        i. Nachname
        i. Geburtsdatum
* Fünf

^D
```

Nachdem ich die Eingabe per `^D` (`[Strg]+[D]`) abgebrochen habe, erscheint prompt der konvertierte LaTeX-Code im Terminal:

```
\begin{itemize}
\itemsep1pt\parskip0pt\parsep0pt
\item
  Eins
\item
  Zwei
\item
  Drei

  \begin{enumerate}
  \def\labelenumi{\arabic{enumi}.}
  \itemsep1pt\parskip0pt\parsep0pt
  \item
    alpha
  \item
    beta
  \item
    gamma
  \end{enumerate}
\item
  Vier

  \begin{enumerate}
  \def\labelenumi{\arabic{enumi}.}
  \itemsep1pt\parskip0pt\parsep0pt
  \item
    erstens
  \item
    zweitens

    \begin{enumerate}
    \def\labelenumii{\roman{enumii}.}
    \itemsep1pt\parskip0pt\parsep0pt
    \item
      Vorname
    \item
      Nachname
    \item
      Geburtsdatum
    \end{enumerate}
  \end{enumerate}
\item
```

```
  Fünf
\end{itemize}
```

Weiter mit HTML

Könnte ich auch herausfinden, wie das als HTML rauskommt? Klar, natürlich:

```
$> pandoc -f markdown -t html
* Eins
* Zwei
* Drei
    1. alpha
    1. beta
    1. gamma
* Vier
    1. erstens
    1. zweitens
        i. Vorname
        i. Nachname
        i. Geburtsdatum
* Fünf

^D

<ul>
<li>Eins</li>
<li>Zwei</li>
<li>Drei
<ol style="list-style-type: decimal">
<li>alpha</li>
<li>beta</li>
<li>gamma</li>
</ol></li>
<li>Vier
<ol style="list-style-type: decimal">
<li>erstens</li>
<li>zweitens
<ol style="list-style-type: lower-roman">
<li>Vorname</li>
<li>Nachname</li>
<li>Geburtsdatum</li>
</ol></li>
</ol></li>
<li>Fünf</li>
</ul>
```

Wie geht eine Tabelle?

Und wie schreibt `pandoc` eine einfache Tabelle als LaTeX?

```
$> pandoc -f markdown -t latex

Eine 'faule' Tabelle|Zweite Spalte|Dritte, rechtsbündig
:-------------------|-------------|-------------------:
Äpfel|€ 2.45|23.45 $
Birnen|€ 122.68|1.- $
Pflaumen|€ 13|55.91 $

Table: links=linksbündig; mitte=default; rechts=rechtsbündig

^D

\begin{longtable}[c]{@{}llr@{}}
\toprule\addlinespace
Eine `faule' Tabelle & Zweite Spalte & Dritte, rechtsbündig
\\\addlinespace
\midrule\endhead
Äpfel & \euro{} 2.45 & 23.45 \$
\\\addlinespace
Birnen & \euro{} 122.68 & 1.- \$
\\\addlinespace
Pflaumen & \euro{} 13 & 55.91 \$
\\\addlinespace
\bottomrule
\addlinespace
\caption{links=linksbündig; mitte=default; rechts=rechtsbündig}
\end{longtable}
```

Konvertierung dieses Dokuments

Das Konvertieren von Markdown-Text in andere Formate per `pandoc` hat bereits der Einführungs-Artikel behandelt.

Vertiefung: Literaturliste automatisch erzeugen und einbinden

`pandoc` kann in seine Ausgabe-Dokumente automatisch ein Literaturverzeichnis einbinden. Hierzu ruft man die Kommandozeile mit den folgenden ergänzenden Parametern auf:

```
  --filter=pandoc-citeproc          \
  --csl=mhra.csl                    \
  --biblio=mybiblio.bib             \
```

`pandoc-citeproc`

`pandoc-citeproc` ist ein Zusatz-Programm für das `pandoc`-Hauptprogramm, welches dafür sorgt, dass Literaturverweise in den konvertierten Ziel-Dokumenten landen und dort richtig referenziert sind.

Stil-Datei: `.csl`

Die Datei `*.csl` legt dabei den exakten Stil und das Format der Literaturverweise fest. Solche Stil-Dateien gibt es ~~dutzendfach~~ ~~zu hunderten~~ ~~zu tausenden~~ aber-tausende im Internet[6] – beinahe jeder Verlag und jede Organisation, die regelmäßig wissenschaftliche Arbeiten publiziert, pflegt einen haus-eigenen Standard oder Stil, dessen Befolgung sie von ihren Autoren verlangen. Im vorliegenden Fall habe ich die *ieee.csl* der *Institutes of Electrical and Electronics Engineers* von folgender URL heruntergeladen:

- https://raw.githubusercontent.com/citation-style-language/styles/master/ieee.csl

Bibliotheks-Datenbank: `.bib`

Die Datei `*.bib` enthält in einem standardisierten Format die Informationen zu meinen Literatur-Quellen. Für das vorliegende Dokument habe ich die Datei *my-biblio.bib* selbst erstellt, und mit einigem (nicht ganz ernst zu nehmenden) Inhalt befüllt:

```
@Book{ref1,
  author="Thales von Milet",
  title="Doppelwinkel-Funktionen",
  url="http://de.wikipedia.org/wiki/Formelsammlung_Trigonometrie#Doppelwinkelfunktionen",
  year="600 v.Chr.",
  address="Milet, Kleinasien",
  publisher="Wikipedia"
}

@Article{ref2,
  author="OStR Dr. math.nat. Oster",
  title="Unterrichtsmaterialen für Klasse 9 (Mittelstufe)",
```

[6] z.B. hier: http://www.zotero.org/styles/

```
  year="1969",
  journal="Journal of Generic Studies",
  volume="9",
  pages="33-34"
  address="Große Kreisstadt Calw",
  url="",
  editor=""
}

@InCollection{ref3,
  author="Elvis Presley, Madonna and Pink Floyd",
  title="Kombinatorik Hypergeometrischer Verteilungen",
  booktitle="Wiederholungslose Auswahlprobleme",
  editor="Cleopatra, Königin von Ägypten",
  publisher="Steintafeln Moses GmbH & Co. KG",
  address="Gizeh",
  year="30 v.Chr."
}

@Article{ref4,
  author="Jane Doz",
  title="Why All Men Suck",
  year="2006",
  journal="Journal of Gender Studies",
  volume="6",
  pages="33-34"
}
```

> *Anmerkung: Unvollständig befüllte `*.bib`-Datenbanken führen selbstverständlich zu unvollständigen Einträgen im automatisch erstellten Literaturverzeichnis...*

Ausgabeformate mit Literaturverzeichnis

Automatisch erzeugte Literatur-Verweise mit Gesamt-Verzeichnis können nach korrektem `pandoc`-Lauf in LaTeX-, PDF-, ODT-, HTML-, ConTeXt-, EPUB- und DOCX-Output erscheinen. Andere Ausgabe-Formate unterstützen Literaturverzeichnisse nicht.

(Allerdings wird dieser Zusammenhang durch [4] entschieden bestritten!)

Die Markierungen `@ref1`, `@ref2`, `@ref3` und `@ref4` habe ich bei der Diskussion der LaTeX-Formel-Erstellung im Markdown-Quelltext für den Abschnitt *"Umgang mit Bibliographien"* als Anker-Punkte benutzt. (Bitte nochmals zurückblättern, um den entsprechenden Markdown-Code zu betrachten). Das Ergebnis dieser automatisch erzeugten Verweise ist jetzt im über-nächsten Kapitel zu

bestaunen. `pandoc` platziert die Literatur-Verweise immer ganz am Ende der Dokumente.

EBook-Publishing bei Leanpub.com

- Ein Internet-Verlag, der eine modifizierte `markdown+pandoc`-Toolchain verwendet und exzellente Konditionen sowohl für Autoren wie auch für Leser/Käufer bietet. Die dort erhältlichen Formate sind PDF, MOBI und EPUB: **Leanpub.com**[7]

- *"PDF-KungFoo with Ghostscript & Co."* (englisch) – mein EBook bei Leanpub als 'work in progress' mit Minimal-Preis $US 0,00: **https://leanpub.com/pdfkungfoo**
  Bitte trotzdem einen selbstgewählten, freiwilligen Betrag bezahlen. Er geht, nach Abzug von 10% plus 50 Cent pro Buchkauf (das ist der Verlags-Anteil) als Spende an die EFF *(Electronic Frontier Foundation)*.

- *"PDF-KungFoo Workshop bei der Ubucon 2013"* (deutsch) – ein kollaborativ erstelltes EBook mit Minimal-Preis $US 0,00:
  https://leanpub.com/pdfkungfoo-ws1-deu Bitte trotzdem einen selbstgewählten, freiwilligen Betrag bezahlen. Er geht, nach Abzug von 10% plus 50 Cent pro Buchkauf (das ist der Verlags-Anteil) als Spende an die EFF *(Electronic Frontier Foundation)*.

Verwendete Literatur

Die unten folgenden Literatur-Angaben sind natürlich "Dummys". Sie dienen als Platzhalter. Sie sollen lediglich einen Eindruck vermittel, wie die im Markdown-Quelltext referenzierten Zitate und Bibliographie-Angaben (siehe vorletztes Kapitel zu diesem Thema) im fertigen Buch wirken würden.

Der Stil der erzeugten Bibliographie hängt stark davon ab, welche CSL-Datei man via Pandoc aufruft.

Leser, die aufgrund der Teilnahme an einem meiner Workshops Zugriff auf den Markdown-Quellcode haben, sollten mit den verschiedenen möglichen Zitier-Stilen spielen. In folgendem GitHub-Repository findet man weiteres Material:

- **https://github.com/KurtPfeifle/pandoc-csl-testing**

[7] https://leanpub.com/authors/

[1] T. von Milet, *Doppelwinkel-Funktionen*. Milet, Kleinasien: Wikipedia, 600 v.Chr.

[2] O. D. math.nat. Oster, "Unterrichtsmaterialen für Klasse 9 (Mittelstufe)," *Journal of Generic Studies*, vol. 9, pp. 33–34, 1969.

[3] M. Elvis Presley and P. Floyd, "Kombinatorik Hypergeometrischer Verteilungen," in *Wiederholungslose Auswahlprobleme*, K. von Ä. Cleopatra, Ed. Gizeh: Steintafeln Moses GmbH & Co. KG, 30 v.Chr.

[4] J. Doz, "Why All Men Suck," *Journal of Gender Studies*, vol. 6, pp. 33–34, 2006.