

# L<sup>A</sup>T<sub>E</sub>X für Naturwissenschaftler

Ansprechender Text- und Formelsatz von Abschlussarbeiten

## 10. Praktische Aspekte: Quellcode-Formatierung, Dateiorganisation, Versionierung, Automatisierung

Albert-Ludwigs-Universität Freiburg



**UNI  
FREIBURG**

Dr. Till Biskup

Institut für Physikalische Chemie  
Albert-Ludwigs-Universität Freiburg  
Sommersemester 2018



- 🔑 Wissenschaftler tragen eine große Verantwortung: Nachvollziehbarkeit ihrer Arbeit ist das oberste Gebot.
- 🔑 Quellcode sollte übersichtlich formatiert werden. Er wird viel häufiger gelesen als geschrieben.
- 🔑 Die Organisation der Dateien einer Abschlussarbeit dient dazu, die Nachvollziehbarkeit sicherzustellen.
- 🔑 Eine Versionsverwaltung bewahrt die Übersichtlichkeit und hilft beim Arbeiten an unterschiedlichen Rechnern.
- 🔑 Viele Prozessschritte von  $\LaTeX$  sind automatisierbar. Das erspart Arbeit und sorgt für Konsistenz.

- ▶ Wissenschaftler tragen große Verantwortung.
    - Nachvollziehbarkeit Voraussetzung für Wissenschaftlichkeit
  - ▶ sollte sich in der wissenschaftlichen Arbeit widerspiegeln
    - Umgang mit Messungen
    - Organisation der Datenablage
    - Nachvollziehbarkeit der Datenverarbeitung sicherstellen
    - Dokumentation der Ergebnisse und *aller* Schritte dorthin
  - ▶ Das Thema an sich ist sehr groß.
    - geht weit über den Umgang mit  $\text{\LaTeX}$  hinaus
    - hier nur ganz wenige, wesentliche Aspekte
- ☞ weitere Details in der Vorlesung  
**Programmierkonzepte in der Physikalischen Chemie**

Quellcode-Formatierung

Organisation der Dateien

Versionsverwaltung und verteiltes Arbeiten

Automatisierung

## These

Quellcode wird viel häufiger gelesen als geschrieben.

- ▶ Lesbarkeit hat weitreichende Auswirkungen.
  - Arbeitserleichterung und Beschleunigung
  - erleichterte Nachvollziehbarkeit
- ▶ Lesbarkeit hat unterschiedliche Adressaten.
  - der ursprüngliche Autor – spätestens nach Tagen
  - nachfolgende „Generationen“
- Quellcode offenbart wie wenig Anderes die Arbeitsweise.
- Bei  $\text{\LaTeX}$  ist Vieles tolerierbar – bei Programmen nicht ...

- ▶ Länge von Zeilen
  - ähnliche Kriterien wie beim Textsatz
  - Lesbarkeit entscheidend von der Länge abhängig
- ▶ horizontaler und vertikaler Leerraum
  - Strukturierung
  - Zusammengehörendes beisammen lassen
  - Unterschiedliches räumlich klar trennen
- ▶ übersichtliche Befehlsstrukturen
  - ggf. Umgebungen der Befehlsvariante vorziehen
- ▶ sprechende Namen für eigene Befehle
  - semantische Textauszeichnung
  - Die Bedeutung eines Befehls sollte offensichtlich sein.

- ▶ Zeilen sollten nicht zu lang sein.
  - Konkrete Zeichenzahlen hängen vom Kontext ab.
  - Das harte Limit von 80 Zeichen ist nicht mehr zeitgemäß.
  - Mehr als 140 Zeichen pro Zeile sind nie sinnvoll.
- ▶ Die meisten Editoren brechen automatisch Zeilen um.
  - *soft wrap*
  - Manuell eingefügte Zeilenumbrüche sind eher mühsam, insbesondere wenn sich der Text aktiv ändert.
  - Ausnahme: Kommentare  
(werden oft nicht automatisch umgebrochen)
- ▶  $\text{\LaTeX}$ 
  - Zeilenumbrüche sind (meistens) egal.
  - Absätze durch Leerzeilen
  - Zeilenumbrüche im Textsatz selten notwendig

- ▶ Bedeutung
  - wesentliches Element der Gliederung von Quellcode
  - sollte mit dem Inhalt korrespondieren
  - Konsistenz hilft beim Verständnis des Quellcodes.
  
- ▶ Textsatz mit  $\text{\LaTeX}$ 
  - Beliebig viele Leerzeichen werden als eines interpretiert.
  - Beliebig viele Leerzeilen markieren einen Absatz.
  - harter Zeilenumbruch wird als Leerzeichen interpretiert
  
- ▶ mathematischer Formelsatz mit  $\text{\LaTeX}$ 
  - Leerzeilen nicht erlaubt, deshalb auskommentieren
  - Zeilenumbrüche nicht als Formelumbruch interpretiert
  - Tipp: Formelteile übersichtlich im Quellcode umbrechen
  - Umbrüche durch auskommentierte Leerzeilen markieren

- ▶ Besonderheit von  $\text{\LaTeX}$ 
  - viele Befehle auch als Umgebung verfügbar
  - Befehle tendenziell im laufenden Text
  - Umgebungen eher für ganze Absätze
- ▶ Nachteile der Befehlsvariante für längere Inhalte
  - schließende geschweifte Klammer wenig prominent
- ▶ Tipps aus der eigenen Praxis
  - für längere Inhalte Umgebungen bevorzugen
  - in jedem Fall für übersichtliche Formatierung sorgen (Leerzeilen, ggf. Einrückung)
- ☞ Die Umgebung/der Befehl `frame` aus der `beamer`-Klasse ist ein besonders gutes Beispiel.

### Listing 1: Grundstruktur eines Rahmens (Folie) in Befehlsform

```
\frame[Optionen]{  
\frametitle{Titel}  
\framesubtitle{Untertitel}  
...  
}
```

- ▶ Befehlsvariante
  - widerspricht ein wenig der  $\text{\LaTeX}$ -Philosophie
  - Ein `frame` hat normalerweise ausführlicheren Inhalt.
- ▶ Ende des Befehls nur durch geschweifte Klammer
  - sollte *immer* auf eigener Zeile stehen
  - leicht übersehbar
  - nachfolgender vertikaler Leerraum besonders wichtig

### Listing 2: Grundstruktur eines Rahmens (Folie) in Umgebungsform (1)

```
\begin{frame}[Optionen]{Titel}{Untertitel}  
...  
\end{frame}
```

---

### Listing 3: Grundstruktur eines Rahmens (Folie) in Umgebungsform (2)

```
\begin{frame}[Optionen]  
\frametitle{Titel}  
\framesubtitle{Untertitel}  
...  
\end{frame}
```

---

☞ Die dritte Form ist die übersichtlichste Variante.

- ▶ Zu kurze Befehlsnamen sind selten eine gute Idee.
  - transportieren wenig Bedeutung
  - Gefahr der Kollision mit anderen Befehlen
  - Ausnahme: mathematische Konstanten
  - klarer Kontext, der aber Wissen voraussetzt
  
- ▶ semantische Textauszeichnung
  - Formatierung im Textsatz transportiert Bedeutung.
  - Befehlsnamen übertragen Bedeutung in Quellcode
  - Treffende Namen lassen sich leicht merken.
  - Ziel: Quellcode soll Verständnis ermöglichen.
  
- ▶ Vorbilder aus dem Standard-Repertoire von  $\text{\LaTeX}$ 
  - `\tableofcontents` – Bedeutung sofort klar
  - griechische Buchstaben – intuitiv und leicht zu merken

- ▶ nur in der Präambel
  - im Text nur kurzfristig zur Fehlersuche, um Bereiche auszukommentieren
  - in Verbindung mit Versionierung (s.u.) ist keine „Sicherung“ von Textteilen notwendig
- ▶ Funktion und Bedeutung
  - Gruppierung von Einstellungen in der Präambel
  - kurze Erklärung eher kryptischer  $\text{\LaTeX}$ -Befehle
- ▶ sparsam
  - meist nur einzeilig
  - prägnant
  - sollten immer konsistent mit dem zugehörigen Code sein
  - je weniger, desto weniger falsche Kommentare

- ▶ Daten einer Abschlussarbeit
  - Rohdaten
  - verarbeitete Daten
  - entwickelte/verwendete Auswerteroutinen
  - Literatur
  - schriftliche Abschlussarbeit
  - ggf. Abschlussvortrag
- ▶ Voraussetzungen für die Nachvollziehbarkeit
  - Zugriff auf Rohdaten und verarbeitete Daten
  - detaillierte Dokumentation aller Verarbeitungsschritte
- ☛ Die Organisation der Dateien betrifft weitaus mehr als die  $\LaTeX$ -Quellen zur Arbeit.

- ▶ generelle Aspekte
  - Unterverzeichnisse für größere Übersichtlichkeit
  - Auslagern der Header und Inhalte in einzelne Dateien
  - Einbinden über `\input` bzw. `\include`
  
- ▶ Benennung des Hauptdokuments
  - allgemeines Schema: `Typ-NachnameVorname`
  - Typ: BSc, MSc (je nach akademischem Grad)
  - erleichtert die Zuordnung außerhalb des engen Kontextes
  - für den Vortrag entsprechend Dateinamen erweitern
  
- ▶ weitere Tipps
  - Abbildungsverzeichnis evtl. auf gleicher Ebene wie Abschlussarbeit und Vortrag
  - identische Abbildungen bei beiden verwendbar

- ▶ darüber hinausgehende Aspekte
  - Organisation der gesamten Dateien einer Abschlussarbeit
  - Ziel: Nachvollziehbarkeit
- ▶ Nachvollziehbarkeit durch andere
  - Die Zeitskala eines Studenten im AK ist begrenzt.
  - Wissenschaft findet auf viel längeren Zeitskalen statt.
  - Nachvollziehbarkeit erfordert Unabhängigkeit von Personen.
- ▶ Schema
  - erst einmal AK-intern
  - entspringt durchaus der Erfahrung
  - muss sich praktisch bewähren und ggf. angepasst werden
- ☞ Die Verantwortung jedes Einzelnen ist es, das gegebene Schema kritisch zu hinterfragen und umzusetzen.

### Experimentelle/theoretische Arbeiten

- ▶ Rohdaten
  - eigentliche Messdaten
  - Theoriewerke (DFT-Rechnungen etc.): Ausgabedateien
- ▶ Auswertungen
  - für die Datenverarbeitung verwendete Skripte/Programme
  - bei der Auswertung erzeugte abgeleitete Daten

### Arbeiten mit Schwerpunkt auf Programmentwicklung

- ▶ entwickelte Programme
  - sauber dokumentiert
  - idealerweise in einer Versionsverwaltung

- ▶ **Abbildungen**
  - alle erstellten Abbildungen in *bearbeitbarem Format*
  - nach Möglichkeit Vektorgrafiken
- ▶ **Literatur**
  - PDF-Dateien der für die Arbeit verwendeten Literatur
  - Webseiten: PDF-Dokument der zitierten Version, Datum
  - bei  $\text{BIB}\text{T}_\text{E}\text{X}$ : Dateiname ggf. identisch mit  $\text{BIB}\text{T}_\text{E}\text{X}$ -Schlüssel
- ▶ **These**
  - eigentliche Arbeit als PDF-Dokument
  - bei  $\text{L}\text{A}\text{T}_\text{E}\text{X}$ : gesamter Ordner mit allen Dateien
- ▶ **Vortrag**
  - Abschlussvortrag als PDF-Dokument
  - bei  $\text{L}\text{A}\text{T}_\text{E}\text{X}$ : gesamter Ordner mit allen Dateien

- ▶ typische Situation
  - Arbeiten an einer Arbeit an mehr als einem Rechner
  - Daten synchron zu halten ist mit Aufwand verbunden.
- ▶ typische Lösungsstrategien
  - Daten werden auf dem USB-Stick transferiert.
  - $\text{\LaTeX}$ -Dateien werden per Email verschickt.
  - Große Datenmengen werden in irgendeine Cloud gelegt.
- ▶ Probleme typischer Lösungsstrategien
  - Synchronisation ist nicht trivial.
  - Cloud-Lösungen sind meist illegal.
- ▶ Lösungsstrategien
  - Versionsverwaltung u.a. zur Synchronisation
  - Nutzung der lokalen Infrastruktur

- ▶ Gründe für eine Versionsverwaltung
  - Nachvollziehbarkeit (zusammen mit Versionsnummern o.ä.)
  - erlaubt verteiltes Arbeiten
  - erleichtert Zusammenarbeiten an einem Projekt
- ▶ git – *everything is local*
  - verteiltes Versionsverwaltungssystem
  - unabhängig von jeglicher Server-Infrastruktur
  - lässt sich in einzelnen Verzeichnissen nutzen
- ▶ git vs. gitlab/github
  - git ist das eigentliche Versionsverwaltungssystem.
  - gitlab/github sind Plattformen für den webbasierten Zugriff.
  - In der PC ist eine lokale gitlab-Installation verfügbar.
- ☞ Für die sinnvolle Entwicklung von Programmen ist eine Versionsverwaltung unumgänglich.

### verfügbare Möglichkeiten

- ▶ von außen erreichbare git-Repositories
  - auf dem (laufenden) Arbeitsplatzrechner (via SSH)
  - gitlab-Installation (in der PC vorhanden)
  - Kostenlose Accounts bei github/gitlab sind *nicht* geeignet, da sie keine privaten Repositories erlauben.
- ▶ von außen erreichbare Speicherlaufwerke
  - zentraler Speicher via UniAccount
  - zentrales Laufwerk des Arbeitskreises (in der PC verfügbar)
- ☛ AK-intern: vom Arbeitsplatz aus nur auf dem NAS arbeiten (Alles andere wird nach dem Ausscheiden gelöscht.)

### Satz

Computer sind dumm, aber fleißig.

### Gründe für Automatisierung

- ▶ kommt der menschlichen Bequemlichkeit entgegen
- ▶ erlaubt Fokussierung auf die wichtigen Dinge
  - Auswertung ist selten automatisierbar.
- ▶ sorgt für Konsistenz
  - erfordert anderenfalls große Disziplin und Planung
- ▶ erlaubt, bei Fehlern an einer Stelle zu ändern
  - Konsistenz, s.o.

- ▶ Hauptdokument festlegen
  - bei Aufteilung in mehrere Dokumente
  - wird von quasi allen  $\text{\LaTeX}$ -IDEs unterstützt
- ▶ Bibliographie erstellen
  - normaler Ablauf:  $\text{\LaTeX}$ , biber/ $\text{\BibTeX}$ ,  $\text{\LaTeX}$ ,  $\text{\LaTeX}$
  - gilt ähnlich für Indices etc.
- ▶ Präsentationen: unterschiedliche Modi
  - manuell: Optionen an die Dokumentklasse übergeben
  - Ziel: keine Änderungen im Quellcode
  - automatisch: Optionen beim  $\text{\LaTeX}$ -Aufruf übergeben
- ▶ Aktualisierung des Dokuments automatisieren
  - Aufruf externer Programme in der richtigen Reihenfolge

- ▶ IDEs (Texmaker, Kile, ...)
  - Festlegen des Hauptdokuments
  - Einstellung der „richtigen“ Werkzeuge
  - Kompilierung etc. aus der grafischen Oberfläche
  
- ▶ Makefile
  - aus der Welt der Programmierung
  - unter unixoiden Systemen
  - bei den Vorlagen für Arbeit/Vortrag mitgeliefert
  
- ▶ latexmk
  - Kommandozeilenprogramm
  - plattformübergreifend
  - sorgt für automatische Aktualisierung



### Listing 4: Makefile für die Abschlussarbeit

```
LATEXBASEFILE = BSc-WurstHans
```

```
pdf:
```

```
  pdflatex $(LATEXBASEFILE)
```

```
bib:
```

```
  pdflatex $(LATEXBASEFILE)
```

```
  biber $(LATEXBASEFILE)
```

```
  pdflatex $(LATEXBASEFILE)
```

```
  pdflatex $(LATEXBASEFILE)
```

```
clean:
```

```
  rm -f *aux *lo? *xml *out *toc *bbl *bcf *blg *.mtc* *maf
```

```
  rm -f Includes/*aux
```

- ▶ genereller Aufruf
  - `make`
  - ggf. zusätzliche Option („Ziel“)
- ▶ Funktionalität der Variante der Vorlagen
  - Hauptdatei festgelegt
  - `make` kompiliert das Dokument (identisch zu `make pdf`)
  - `make bib` erstellt (zusätzlich) das Literaturverzeichnis
  - `make clean` löscht temporäre Dateien
  - `make trans` für druckbare Folien des Vortrags
- ▶ notwendige Anpassungen
  - Name des Hauptdokuments
  - ggf. Bibliographie-Programm (`biber` oder `BIBTEX`)



- 🔑 Wissenschaftler tragen eine große Verantwortung: Nachvollziehbarkeit ihrer Arbeit ist das oberste Gebot.
- 🔑 Quellcode sollte übersichtlich formatiert werden. Er wird viel häufiger gelesen als geschrieben.
- 🔑 Die Organisation der Dateien einer Abschlussarbeit dient dazu, die Nachvollziehbarkeit sicherzustellen.
- 🔑 Eine Versionsverwaltung bewahrt die Übersichtlichkeit und hilft beim Arbeiten an unterschiedlichen Rechnern.
- 🔑 Viele Prozessschritte von  $\LaTeX$  sind automatisierbar. Das erspart Arbeit und sorgt für Konsistenz.

## Ein paar Fragen als Inspiration:

- ▶ Waren die Themen verständlich?
  - ▶ Hat die Thematik für Euch eine Relevanz?
  - ▶ Entsprach der Kurs Euren Vorstellungen?
  - ▶ Was könnte man besser machen?
  - ▶ Was fehlte?
  - ▶ Was war zu viel?
  - ▶ ...
- ☞ Sollte ein solcher Kurs regelmäßig angeboten werden?