LaTeX, beamer, tikz und Co.

18. Dokumente auf mehrere Dateien verteilt

Thomas Worsch

Fakultät für Informatik
Karlsruher Institut für Technologie

Wintersemester 2016/2017

Wozu Dokument auf mehrere Dateien verteilen?

Wozu Dokument auf mehrere Dateien verteilen?

- Wiederverwendung von Teilen
- mehr Übersicht (?)
- separate Übersetzung einzelner Teile (?)
  - Teile als eigenständige Dokumente
- Übersetzung von Teilen nur bei Bedarf (?)

Kommando `\input`

Der `include`-Mechanismus

Pakete `graphics` und `graphicx`

TikZ library `external`

Paket `standalone`

Überblick

Einfache Methode

- Syntax `\input{`⟨*Dateiname*⟩`}`
  - geschweifte Klammern *wichtig*
- ⟨*Dateiname*⟩
  - mit Endung `.tex`:
    - wird so genommen
  - ohne Endung `.tex`:
    - erst wird ⟨*Dateiname*⟩ gesucht, falls nicht zu finden,
    - wird anschließend nach ⟨*Dateiname*⟩`.tex` gesucht
- Bedeutung:
  - LaTeX verhält sich ungefähr so, also stände an Stelle von `\input{`⟨*Dateiname*⟩`}` der Inhalt der Datei
  - aber nicht ganz: siehe `t-input.tex`, `t-input-2.tex`
- *Tipp*: siehe nächste Folie

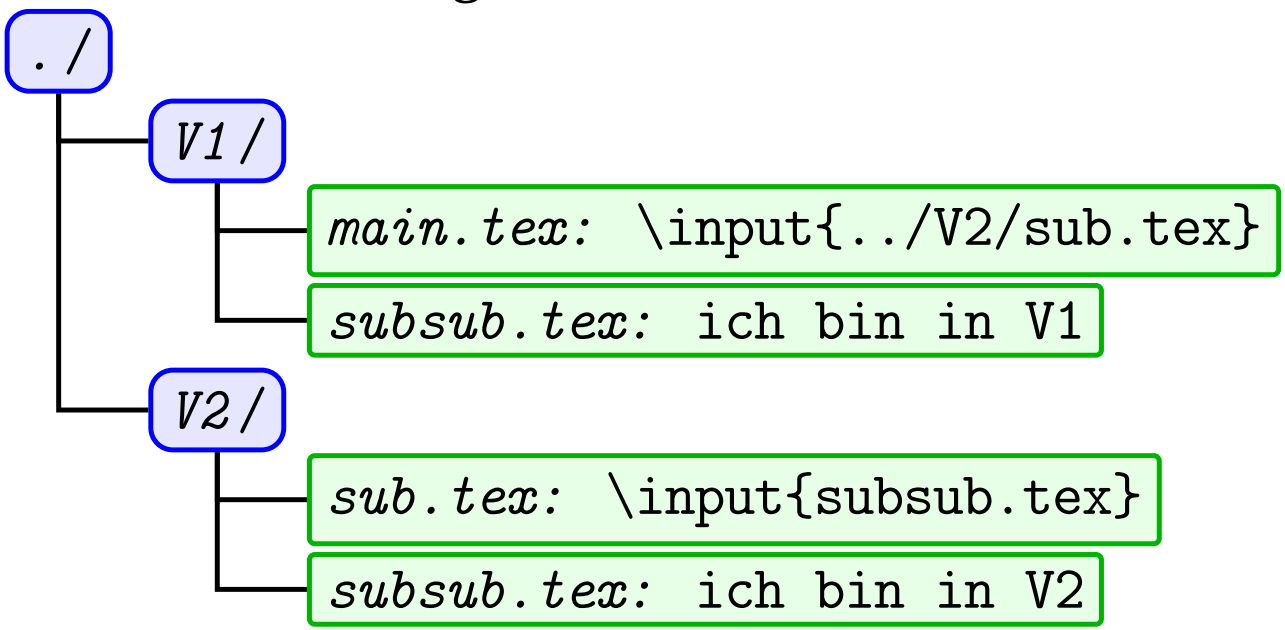
Kommando \endinput

- kann an jeder Stelle in einer Datei stehen
- Wirkung: beendet das Einlesen der Datei durch LaTeX
  - Text danach wird „ignoriert“

- *Tipp*: Am Ende jeder Datei einfügen
  - hat auch den gleichen Effekt wie ein % am Dateiende

Pfade für Dateinamen bei \input

- funktionieren
- aber Vorsicht in folgender Sitation:



  - bei `pdflatex main.tex` in `V1`
    wird auch `subsub.tex` aus `V1` gelesen!
    - `\input{../V2/subsub.tex}` würde das aus `V2` lesen
- Alternative: Paket `import`

Überblick

Kommando \include

- Syntax: \include{⟨*Dateiname*⟩}
  - im Dokumentenrumpf
- Bedeutung ähnlich wie \input{⟨*Dateiname*⟩}
  - sofern keine \includeonly Kommandos vorhanden
    sonst: siehe nächst Folie
  - aber *es wird davor und danach eine neue Seite begonnen*
- es wird auch eine separate .aux-Datei erzeugt
- \include können nicht „geschachtelt“ werden

- wenn ⟨*Dateiname*⟩ nicht existiert
  - erzeugt \include{⟨*Dateiname*⟩} nur eine *Warnung*
  - erzeugt \input{⟨*Dateiname*⟩} einen *Fehler*

Kommando \includeonly

- Syntax: \includeonly{⟨*Dateinamen*⟩}
  - in der Präambel
  - oder davor (geht jedenfalls zur Zeit)
- ⟨*Dateinamen*⟩
  - falls mehrere, durch Kommata getrennt
  - für jeden Dateinamen ein \include-Kommando im Rumpf

- Bedeutung: Wenn \includeonly in der Präambel benutzt
  - dann nur entsprechende \include Kommandos „*aktiv*“
  - die anderen Inhalte werden *ignoriert*
    - aber Seitenzahlen, Labels, usw. bleiben erhalten!

  ⇝ kürzere Verarbeitungszeit

Beispielanwendung (1)

```
\documentclass{article}
\includeonly{part-2}
\begin{document}
\include{part-1}
\include{part-2}
\include{part-3}
\end{document}
```

Beispielanwendung (1)

```
\includeonly{part-2}
\documentclass{article}
\begin{document}
\include{part-1}
\include{part-2}
\include{part-3}
\end{document}
```

Beobachtung: es funktioniert auch, wenn die ersten beiden Zeilen vertauscht sind …

Beispielanwendung (2)

- Datei `all.tex`

```
\documentclass{article}
\begin{document}
\include{part-1}
\include{part-2}
\include{part-3}
\end{document}
```

- Datei `part-2-alone.tex`

```
\includeonly{part-2}
\input{all}
```

- Vorteil: `all.tex` muss nicht geändert werden

Überblick

Einbinden von Bildern in „Fremdformaten“

- benutzbare Bildformate:
  - hängt vom benutzten LaTeX ab
  - für `pdflatex`: Pdf-, Jpeg-, PNG-Bilder
- Unterschied zwischen `graphics` und `graphicx`
  - funktional: keiner
  - Benutzerschnittstelle: ich bevorzuge `graphicx`
  - alles Weitere bezieht sich auf `graphicx`
- wesentliches Kommando
  `\includegraphics[`⟨*opt. key/value pairs*⟩`]{`⟨*filename*⟩`}`

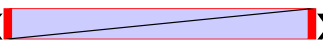
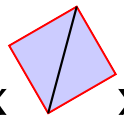
Kommando \includegraphics

mögliche Optionen

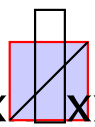
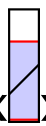
- scale=⟨*Faktor*⟩
- width=⟨*Breite*⟩
- height=⟨*Höhe*⟩
- keepaspectratio
- bb=⟨*llx, lly, urx, ury*⟩ , viewport=⟨*llx, lly, urx, ury*⟩
- clip

\includegraphics: Beispiele

Optionen angegeben

- keine: xxx  xxx
- [width=20mm,height=2mm] xxx  xxx
- [width=20mm,height=2mm,keepaspectratio] xxx  xxx
- [angle=30] xxx  xxx

nachfolgend \framebox{\includegraphics...}}

- [bb=5 0 10 20] xx  xxx
- [bb=5 0 10 20,clip] xxx  xxx
- bb-Angaben sind in bp Einheiten, 1 bp= 1/72 in

(Große) Bilder

- große Bilder kosten Zeit:
  - beim Entwickeln („Programmieren“)
  - beim Verarbeiten („Ausführen“)
- Wie kann man Zeit sparen?
  1. Datei ⟨*bild*⟩`.tex`, die *nur* das Bild enthält
     - reduziert Verarbeitungszeit während der Entwicklung
     - Ergebnis: Pdf-Datei ⟨*bild*⟩`.pdf`

  2a. im Hauptdokument `\includegraphics{`⟨*bild*⟩`.pdf}`
     - reduziert Verarbeitungszeit während der Dokumentenerstellung
     - Wiederverwendbarkeit
     - evtl. Konsistenzprobleme (Fontänderungen, …)

oder 2b. wesentlichen Teil aus ⟨*bild*⟩`.tex` in Hauptdokument einfügen
     - Vorteil: Wiederverwendbarkeit in verschiedenene Dokumenten
       z. B. auch bei verschiedenen Schriften, etc.
     - evtl. Konsistenzprobleme (nachträgliche Änderungen wo? …)

Überblick

TikZ library external: Benutzung

Benutzung im günstigen Fall ganz einfach:

- in der Präambel:
  - `\usetikzlibrary{external}`
  - `\tikzexternalize`
    - schaltet den Mechanismus ein
    - wenn auskommentiert, Dokument „ganz normal“ verarbeitet
- im Dokumentenrumpf:
  - `tikzpicture` Umgebungen wie üblich

- *nur* für `tikzpicture` Umgebungen, nichts anderes

- Tipp: für Erzeugung des finalen Dokuments `\tikzexternalize` *ausschalten*
  - zur Vermeidung unerwünschter Spaces

TikZ library external: Benutzung

Benutzung im günstigen Fall ganz einfach:

- in der Präambel:
  - `\usetikzlibrary{external}`
  - `\tikzexternalize`
    - schaltet den Mechanismus ein
    - wenn auskommentiert, Dokument „ganz normal“ verarbeitet
- im Dokumentenrumpf:
  - `tikzpicture` Umgebungen wie üblich
- *nur* für `tikzpicture` Umgebungen, nichts anderes
- Tipp: für Erzeugung des finalen Dokuments `\tikzexternalize` *ausschalten*
  - zur Vermeidung unerwünschter Spaces

TikZ library external: Mechanismus

für jedes tikzpicture defaultmäßig folgende Aktionen

- Berechnung einer ⟨*md5 sum*⟩
  - falls ⟨*md5 sum*⟩ unverändert:
    - nehme das Pdf-Erzeugnis des letzten Laufes
  - falls ⟨*md5 sum*⟩ geändert (oder erstmals berechnet):
    - erzeuge aus tikzpicture neue Pdf-Datei
- füge Pdf-Datei statt des tikzpicture in Hauptdokument ein

Überblick

Bündel standalone

die einfache Variante:

- reduziert „nur“ Verarbeitungszeit während der Entwicklung von Teilen
  - weniger als tikz external
  - nicht eingeschränkt auf tikzpicture
- für die Benutzung werden *zwei* „Zutaten“ benötigt
  nicht verwechseln:
  - die Dokumentenklasse standalone
  - das LaTeX-Paket standalone

Bündel standalone: einfache Benutzung

- erzeuge *zwei* eigenständige Dokumente (oder mehr …)
  - Hauptdokument main.tex
  - Teildokument sub.tex
    - Inhalt sollte defaultmäßig auf eine Seite passen
- in main.tex:
  - Präambel: \usepackage{standalone}
    - „*early in the document*“
    - lade alle Pakete, die sub.tex braucht
    - oder \usepackage[subpreambles]{standalone}
  - Rumpf: z. B. \input{sub.tex}
- in sub.tex:
  - Präambel: \documentclass{standalone}
    danach Beliebiges
  - Rumpf: beliebiger Inhalt
    „*Avoid empty lines at the begin or end of the document body.*“

Bündel standalone: fortgeschrittene Benutzung

im Rumpf von main.tex:

- \includestandalone[mode=⟨*mode*⟩]{sub.tex}
- für ⟨*mode*⟩ sind unter anderem möglich:
  - tex: benutze die LaTeX-Quelldatei
  - image: benutze die erzeugte „Image“-Datei
  - image|tex: benutze Image, falls vorhanden, sonst Quelle
  - build: erzeuge Image und nutze es
  - buildmissing: erzeuge Image, falls nicht vorhanden
  - buildnew: erzeuge Image, falls Quelle neuer als Image
- statt bei jedem \includestandalone den ⟨*mode*⟩ anzugeben
  - in Präambel: \standaloneconfig{mode=⟨*mode*⟩}
  - für finales Dokument: mode=tex