LaTeX im Studium

LaTeX automatisieren

Jörn Clausen
joern@TechFak.Uni-Bielefeld.DE

Übersicht

- Administrativa, vorbereitende Maßnahmen
- kurze Wiederholung
- Aufruf von LaTeX und Freunden durch `make`

Vorbereitungen

- http://www.TechFak.Uni-Bielefeld.DE/~joern/latex/
- Übungen entpacken:

```
$ tar zxvf uebung1.tar.gz
```

- in `.rcrc`:

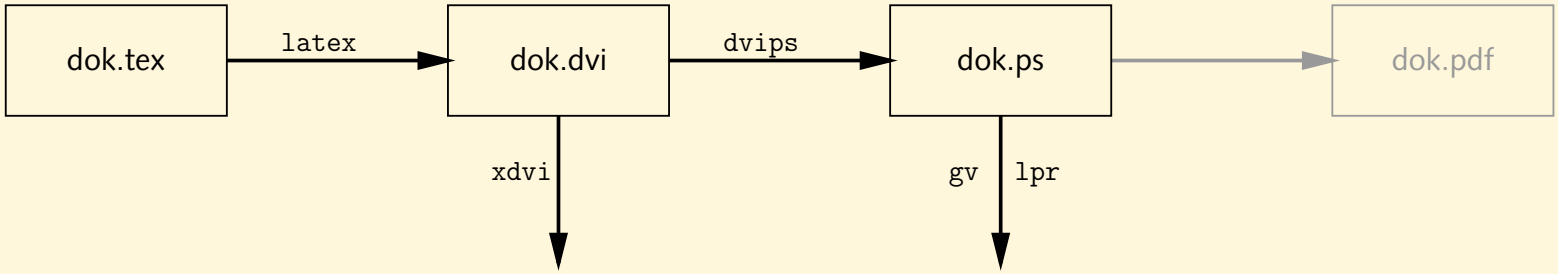
```
RCINFO_ILIST = (acrobat netpbm ...)
```

- in `.emacs`:

```
(require 'tex-site)
(global-font-lock-mode t)
```

ein Lebenslauf

- typische Verarbeitungsschritte:



- weitere Mitspieler: BibTeX, `makeindex`, ...
- Reihenfolge der Übersetzung wichtig
- Problem: Schleifen (LaTeX zweimal aufrufen)

Makefiles

- Ziel-Datei wird aus Quell-Datei(en) erzeugt
- Zustand anhand der *timestamps* erkennbar
- Ziel-Datei neuer als alle Quell-Dateien?

  ja: tue nichts

  nein: übersetze neu
- `Makefile` besteht aus *rules* und *commands*
- rules verknüpfen *targets* und *prerequisites*
- prerequisite einer Regel kann target einer anderen Regel sein

das erste `Makefile`

- als `Makefile` abspeichern:

```
dok.dvi:        dok.tex
        latex dok.tex
```

- lies: „Wenn `dok.dvi` nicht existiert oder älter ist als `dok.tex`, dann führe den angegebenen `latex`-Aufruf aus.“
- auf Tabulatoren achten!
- Aufruf in der Shell:

```
$ make
```

- Probelauf:

```
$ make -n
```

Aufgaben

- Was passiert, wenn Du `make` ein weiteres Mal aufrufst?
- Schreibe eine zweite Regel, um mit Hilfe von `dvips` die dvi-Datei nach PostScript zu konvertieren. Was ist zu beachten?

Reihenfolge der Regeln

- zwei mögliche Lösungen:

```
dok.dvi:        dok.tex
        latex dok.tex

dok.ps:         dok.dvi
        dvips dok.dvi -o dok.ps
```

oder

```
dok.ps:         dok.dvi
        dvips dok.dvi -o dok.ps

dok.dvi:        dok.tex
        latex dok.tex
```

Reihenfolge der Regeln, cont.

- entweder: Ziel beim make-Aufruf explizit angeben:

  ```
  $ make dok.ps
  ```

- oder: *default target*

  ```
  all:            dok.ps
  ```

  als erste Regel im `Makefile`

- Regel hat kein Kommando, stößt nur andere Regeln an

Aufräumen

- Zwischenformate, Hilfsdateien, Log-Dateien, ...
- nehmen Platz weg, werden irgendwann nicht mehr gebraucht
- Löschen „von Hand“ mühsam, gefährlich, ...
- Regeln im `Makefile`:

```
clean:
        rm -f dok.aux dok.log dok.dvi dok.ps
```

- Regel hat keine Abhängigkeit, kann immer ausgeführt werden
- `make` erzeugt Shell-Kommandos:

```
clean:
        rm -f *.aux *.log
```

Aufgaben

- Was passiert, wenn Du „`make clean`“ mehrmals nacheinander aufrufst?
- Ändere die Regel „`clean`“ so ab, daß wirklich nur die Hilfsdateien gelöscht werden, nicht aber die Ergebnisse `dok.dvi` und `dok.ps`. Diese sollen durch ein zweites Target „`realclean`“ gelöscht werden können.
- Stelle sicher, daß durch „`realclean`“ nicht nur die dvi- und die PostScript-Datei, sondern auch die Hilfsdateien gelöscht werden.

Dokumente aufteilen

- große Dokumente auf mehrere Dateien verteilen:
  - Handhabung der Dateien
  - mehrere Autoren schreiben gleichzeitig
  - partielle Übersetzung (Zeit)
  - Wiederverwendbarkeit von Textstücken
- *Stammdatei* wird übersetzt, bindet weitere Dateien ein
- LaTeX-Befehle: `\input{...}` und `\include{...}`
- Dokumentauszüge mit `\includeonly{...}`

Aufgaben

- Die Dateien `multidok.tex`, `part1.tex`, `part2.tex`, `part3.tex` und `part4.tex` sind eine Stammdatei und vier Textdateien. Sieh Dir die Stammdatei an und übersetze sie. Wie sieht das Ergebnis aus? Welche Dateien entstehen bei der Übersetzung?
- Ersetze die `\input`-Befehle durch `\include{...}` und übersetze das Dokument erneut. Was ändert sich? Welche Dateien entstehen jetzt bei der Übersetzung?
- Füge in der Präambel des Stammdokuments die Zeile

  ```
  \includeonly{part3}
  ```

  ein. Was ändert sich nun?

make mit mehreren Quellen

- target hat mehrere prerequisites:

```
multidok.dvi:   multidok.tex part1.tex part2.tex \
                part3.tex part4.tex
        latex multidok.tex
```

- Zeilen durch \ zusammenfassen
- alternativ: mehrere Abhängigkeiten:

```
multidok.dvi:   multidok.tex
        latex multidok.tex

multidok.dvi:   part1.tex part2.tex part3.tex part4.tex
```

- maximal ein target mit Kommando-Zeile

Aufgaben

- Überzeuge Dich davon, daß `multidok.dvi` tatsächlich neu erzeugt wird, wenn das Stammdokument oder eine der eingebundenen Dateien verändert werden.

Makros

- ähnlich zu Variablen:

```
DVIPS = dvips -Ppdf

dok.ps:         dok.dvi
        $(DVIPS) dok.dvi -o dok.ps
```

- alternative Notation: `${DVIPS}`
- Dateinamen sammeln:

```
PARTS = part1.tex part2.tex part3.tex part4.tex

multidok.dvi:   multidok.tex
        latex multidok.tex

multidok.dvi:   $(PARTS)
```

Makros, cont.

- Makros umschreiben:

```
PARTS = part1.tex part2.tex part3.tex part4.tex
AUXS  = $(PARTS:.tex=.aux)
```

- kann nur Suffixe ersetzen:

```
CAPS  = $(PARTS:part=Part)
```

  liefert nicht das erwartete Ergebnis

Aufgaben

- Schreibe die `clean`-Regel so um, daß nicht alle Dateien mit der Endung `.aux` gelöscht werden, sondern nur diejenigen, die tatsächlich entstehen (können).

Kommandos wiederholen

- Bilddateien `dknuth.png`, `llamport.png` und `texmeta.png`
- PNG kann nicht von LaTeX/dvips verarbeitet werden
- zunächst Konversion nach PostScript:

```
dknuth.eps:     dknuth.png
        pngtopnm dknuth.png | pnmtops > dknuth.eps

llamport.eps:   llamport.png
        pngtopnm llamport.png | pnmtops > llamport.eps

texmeta.eps:    texmeta.png
        pngtopnm texmeta.png | pnmtops > texmeta.eps
```

- Änderungen fehleranfällig

Suffix-Regeln

- fasse Regeln zusammen:

```
%.eps:          %.png
        pngtopnm $< | pnmtops > $@
```

- neue Zeichen:

%	Platzhalter
$<	aktuelles prerequisite
$@	aktuelles target

- passende Kommandos durch Einsetzen konkreter Dateinamen
- nicht in allen Varianten von make vorhanden

Aufgaben

- Füge die gezeigte Regel in das Makefile ein. Überzeuge Dich davon, daß Du die PostScript-Dateien mit

  ```
  $ make dknuth.eps llamport.eps texmeta.eps
  ```

  erzeugen kannst.

- Definiere für die PNG- und PostScript-Dateien zwei Makros und verwende sie in den Regeln. Vergiß nicht das target „`clean`“. Kommentiere die entsprechenden Zeilen in „`dok.tex`“ ein und binde die Bilder ein.

- Die EPS-Dateien werden etwas kleiner, wenn man `pnmtops` mit der Option `-rle` aufruft. Definiere Makros für `pngtopnm` und `pnmtops` und verwende sie in der command-Zeile.

Grenzen von Makefiles

- Abhängigkeiten aufgrund der timestamps
- manchmal reicht das nicht:

```
LaTeX Warning: There were undefined references.

LaTeX Warning: Label(s) may have changed.
   Rerun to get cross-references right.
```

- *brute force* Methode: LaTeX immer mehrfach aufrufen

```
dok.dvi:        dok.tex
        latex dok.tex
        latex dok.tex
```

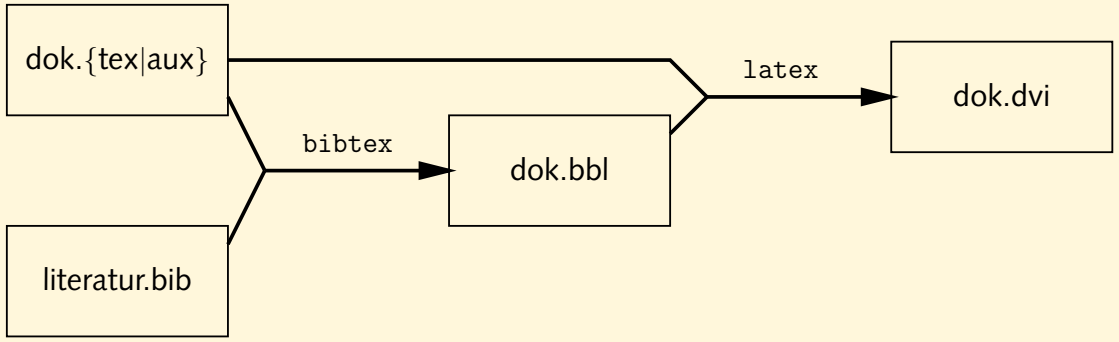
- oder: spezielle targets

Literaturverzeichnisse mit BibTEX

- typische Abfolge, um alle Referenzen aufzulösen:

```
$ latex dok.tex
$ bibtex dok
$ latex dok.tex
$ latex dok.tex
```

- Abhängigkeiten:

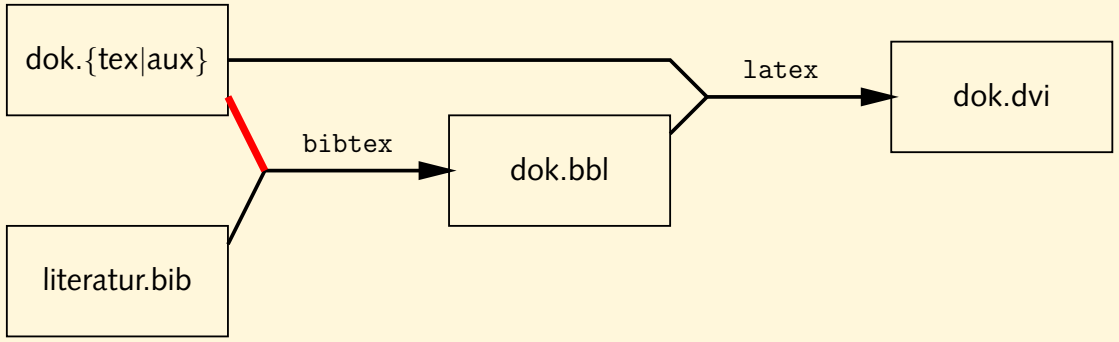


Literaturverzeichnisse mit BibTEX

- typische Abfolge, um alle Referenzen aufzulösen:

```
$ latex dok.tex
$ bibtex dok
$ latex dok.tex
$ latex dok.tex
```

- Abhängigkeiten:



- Abhängigkeit `dok.tex` $\leftrightarrow$ `dok.bbl` nicht modellieren

Aufgabe

- Entferne in „`dok.tex`“ die letzten Kommentarzeichen.
- Erweitere das Makefile so, daß BibTeX aufgerufen wird, falls `dok.bbl` noch nicht existiert oder sich die Literaturdatei `literatur.bib` geändert hat.
- Wie kann man dafür sorgen, daß die Abfolge

  `latex`, `bibtex`, `latex`, `latex`

  zumindest dann korrekt durchlaufen wird, wenn sich `literatur.bib` ändert?
- Füge ein weiteres Zitat in `dok.tex` ein. Kannst Du nur mit Hilfe von `make` eine korrekte dvi-Datei erzeugen?

make tricks

- Makros beim `make`-Aufruf angeben:

```
$ make 'PNMTOPS=pnmtops -rle -scale 0.5'
```

- `make` im Makefile aufrufen:

```
all:
        cd subdir1; make
        cd subdir2; make
```

make tricks

- Makros beim `make`-Aufruf angeben:

```
$ make 'PNMTOPS=pnmtops -rle -scale 0.5'
```

- `make` im Makefile aufrufen:

```
all:
        cd subdir1; make
        cd subdir2; make
```

- BibTeX-Aufruf erzwingen:

```
dok.bbl:        $(FORCEBIBTEX)

bibtex:
        make dok.bbl FORCEBIBTEX=forcebibtex
        latex dok.tex

forcebibtex:
```

Aufgabe

- Füge die gezeigten Regeln in das Makefile ein. Erzwinge den erneuten Aufruf von LaTeX und BibTeX durch

  ```
  $ make bibtex
  ```