LaTeX für Lehrer

Aufgabensammlungen unter Unix/Linux
und andere Hilfsmittel für Lehrer

Problem Col**lect**ion on Un**ix**
(`problectix`)

Rüdiger Beck

16. Juni 2015

Inhaltsverzeichnis

Tabellenverzeichnis

Teil I

Die Distribution — problectix

1 Grundlagen von problectix

1.1 Was ist problectix?

problectix ist eine Sammlung von Werkzeugen (LaTeX-Dokumentklassen, Perl-Scripte, ...) mit denen man als Lehrer mit LaTeX besonders effektiv arbeiten kann.

— Ab hier muss der Text woanders hin —

Jede Aufgabe wird incl. Lösung in einer Datei abgelegt. Alle diese Dateien zusammen bilden die Aufgabensammlung. Soll nun eine Klassenarbeit oder ein Aufgabenblatt erstellt werden, so wird eine Klassenarbeitsdatei bzw. eine Aufgabenblattdatei erstellt, und die gewünschten Aufgaben per Einfügebefehl, der den Dateinamen enthält, eingebunden.

Um den Einfügebefehl der Aufgabe einfach zu ermitteln, können mit einem PERL-script alle Aufgaben-Dateien unterhalb eines Verzeichnisbaums incl. Lösungen in eine html-Seite umgewandelt werden. Dabei liegt die Aufgabe als `*.png`-Bild vor, sowie als Über-/Unterschrift der Einfügebefehl in markierbarer Textform. Der Einfügebefehl kann nun einfach in einen Editor kopiert werden, mit dem die Klassenarbeitsdatei bzw. die Aufgabenblattdatei erstellt wird.

In Zukunft:

Per Script kann eine Aufgabensammung aller Aufgaben erstellt werden (für die Schüler). Da diese Aufgabensammlung ebenfalls in Bild/HTML vorliegt, kann sie per Webserver durchsuchbar gemacht werden.

Da die Erstellten Aufgaben in LaTeX-Code vorliegen, können Sie in eine Versions-Verwaltungs-Software gestellt werden, damit Korrekturen Zentral gesammelt werden können und auch mehrere Personen zugreifen können. Der Zugriff auf den Server kann aus der ganzen Welt erfolgen.

Versionsinformationen von CVS (aber auch git) können können im Dokument ausgegeben werden.

1.2 Die Bestandteile von problectix

problectix besteht aus den folgenden Bestandteilen:

1. LaTeX-Dateien

   - `teacher.cls` : Die Dokumentklasse `teacher` zum Erstellen von Arbeitsblättern, Klassenarbeiten, Prüfungen, Aufgabensammlungen und Stoffverteilungsplänen.
   - `kapack.sty` : Eine Sammlung von LaTeX-Befehlen für die Erstellung von Aufgaben, Aufgabensammlungen.

     `kapack.sty` wird von `teacher.cls` automatisch geladen.

     `kapack.sty` ist nur Funktionsfähig, wenn die Dokumentklasse `teacher` benutzt wird.
   - `teacherpack.sty` : Eine Sammlung von nützlichen LaTeX-Befehlen für immer wieder vorkommende Aufgaben eines Lehrers.

     `teacherpack.sty` wird von `teacher.cls` automatisch geladen.

     `kapack.sty` kann auch mit anderen Dokumentklassen zusammen benutzt werden.

2. LaTeX-Dateien (Experimentell)

   - `bb.cls` : Die Dokumentklasse `bb` (**b**lack**b**oard) ist dazu da ein Tafelbild zu erstellen.
   - `folie.cls` : Eine Dokumentklasse zum Erstellen von *.pdf-Folien.

3. Perl-Scripten

   - `jefflatex` : ein Script zum erzeugen von *.ps bzw. *.pdf aus *.tex-Dateien. Dieses Script kann aus `emacs` heraus aufgerufen werden, und erzeugt auch aus separaten Aufgaben (ohne `\begin{document}`, ...) eine *ps-Datei.
   - `problectix` : Ein Script zum erzeugen von Voransichten von Aufgaben und Aufgabensammlungen die mit `teacher.cls` erstellt wurden.
   - `einmaleins` : Ein Script zum Erzeugen von Einmaleins-Aufgaben in LaTeX-Code.

1.3 Installation von `problectix`

Um `problectix` unter Ubuntu Trusty LTS zu installieren, gehen sie wie folgt vor:

1. Tragen Sie die Paketquelle `trusty-testing` von linuxmuster.net ein (Siehe `http://www.linuxmuster.net/wiki/dokumentation:handbuch:maintenance:repos`).
2. `apt-get update`
3. `apt-get install problectix-teacher-texlive`
4. Installieren sie evtl. noch weitere Pakete.

5. Testen sie die Installation als nomaler `user` mit dem Befehl.

   ```
   user@host:~ # problectix-test
   ```

   Es entsheht ein Verzeichnis `/home/user/problectix-test`, das Beispieldateien in *.pdf und *.ps enthält.

   Der Test kopiert Beispiele (*.tex) nach `/home/user/problectix-test` und führt die Schritte `latex`, `dvips` bzw. `pdflatex` durch.

   Er dauert je nach Rechenleistung einige Minuten!

   Die erzeugten `*.ps` und `*.pdf`-Dateien dienen zur visuellen Kontrolle der korrekten Funktion von `problectix`.

1.4 Konfiguration von `problectix` (Unix/Linux)

1.4.1 Suchpfad für eigne Aufgaben angeben

Wenn in einem LaTeX-Dokument ein `\input{dateiname}`-Befehl auftritt (oder andere Befehle, die auf den Inhalt einer anderen Datei verweisen), dann entscheidet die Umgebungsvariable `TEXINPUTS` in welchen Pfaden nach der Datei `dateiname` gesucht wird.

Den Inhalt der Umgebungsvariablen `TEXINPUTS` kann man sich anzeigen lassen mit:

```
echo $TEXINPUTS
```

Für die Verwendung von `problectix` ist es sinnvoll, ein Verzeichnis einzurichten, in dem sich *alle* Aufgaben und sonstiges Material befindet, dass mit `input` oder ähnlichen Befehlen eingebunden werden kann. Üblicherweise wird dieses Verzeichnis `mytex` genannt. Da dieses Verzeichnis sinnvollerweise mit einer Versionsverwaltung verwaltet werden sollte (z.B. `git` oder `gitolite`) ist es anzuraten `mytex` in das Verzeichnis zu speichern, in dem alle unter `git`-Kontrolle stehenden Projekte gespeichert sind: z.B. `gitolite/mytex`.

Zusätzlich ist es sinnvoll das Verzeichnis `.problectix` einzuschließen, in dem `problectix` benutzerabhängige Einstellungen speichert.

Falls sie Systembetreuer an dem Rechner sind (`root`-Rechte), an dem sie arbeiten, können sie (bei Debian-GNU/Linux) folgendes in `/etc/environement` eintragen.

```
TEXINPUTS="$HOME/Home_auf_Server/gitolite/mytex//::$HOME/.problectix::./figures"
```

Die Umgebungsvariable `TEXINPUTS` wird dann für *alle* Benutzer gesetzt.

Wenn sie keine root-Rechte haben, dann können sie die Datei .bashrc in ihrem Homeverzeichnis anpassen, sodaß sie z.B. folgende Zeile enthält:

```
export TEXINPUTS="$HOME/gitolite/mytex//::$HOME/.problectix::./figures"
```

Die Umgebungsvariable TEXINPUTS wird dann nur für sie gesetzt.

1.4.2 Konfiguration von emacs

Wenn sie Emacs benutzen, installieren Sie das Paket problectix-emacs-texlive. Damit compilieren Sie ihre Texdateien mit:

- F5 zum Erstellen von PostScript (jefflatex -file <Datei> mit dem zugrundeliegenden latex-Befehl)
- <shift>+F5 zum erstellen von PDF's (jefflatex -pdf -file <Datei> und dem zugrundeliegenden pdflatex-Befehl).

Folien können nur mit pdflatex erstellt werden.

Die Funktionstasten werden nur im latex-mode für den Aufruf von jefflatex benutzt. In anderen Modi können sie somit für andere Aufgaben benutzt werden

1.4.3 Konfiguration eines anderen Editors

Richten Sie ihren Editor so ein, dass mit dem Befehl:

```
jefflatex --file <Datei>
```

und mit:

```
jefflatex --pdf --file <Datei>
```

die im Editor gede angezeigte Datei compiliert werden kann.

Teil II

Die Dokumentklasse `teacher.cls`

2 Einführung in die Dokumentklasse `teacher`

Die Dokumentklasse `teacher.cls` dient dazu, Arbeitsblätter, Klassenarbeiten, Prüfungen, Aufgabensammlungen, Stoffverteilungspläne ... zu erstellen. Also alles, was ein Lehrer so an Dokumenten erstellen muss. Die Dokumentklasse wird aufgerufen mit:

```
\documentclass{teacher}
```

Eine Aufgabe, wird mit den Befehlen dieser Dokumentklasse formatiert und in einer separaten Datei abgespeichert. Dann kann die Aufgabe zum Einen in Aufgabenblättern und dazugehörigen Lösungsblättern verwendet werden. Zum Anderen kann sie in Klassenarbeiten und Prüfungen verwendet werden oder in eine umfassende Aufgabensammlung eingebunden werden.

Für die Erstellung typischer Aufgaben sind LaTeX- Befehle vorhanden, die eine einfache Formatierung der Aufgaben ermöglichen, sowie das Zusammenzählen von Punkten, Ein- und Ausblenden der Lösungen, usw. ermöglichen.

3 Die Argumente der Dokumentklasse `teacher`

3.1 Argumente zur Festlegung des Dokumenttyps

Von zentraler Bedeutung sind die Argumente, die es erlauben je nach Dokumenttyp (Klassenarbeit, Prüfung, Arbeitsblatt, ..) Teile der Aufgaben zu verbergen.

Wenn keine dieser Argumente angegeben werden, so wird möglichst *alles* angezeigt.

Folgende Teile einer Aufgabe sind einblend- bzw. ausblendbar:

Datei-/Versionsinfo Dateiname (ohne `.tex`-Erweiterung) sowie cvs-Versionsnummer (bzw git-Datumsangabe)

Aufgabenstellung Die Aufgabe.

Lösungsbereiche Lösungslinien bzw. Lösungskaros, in die die Schüler ihre Lösungen eintragen können.

Korrekturhilfen Punktzahlkasten zum eintragen der Punkte mit Fachangabe

Gruppeninfo Gruppenversion (A,B oder C) der Aufgabe an.

Projektinfo Information ob Projektversion oder die Nicht-Projektversion angezeigt wird.

Lösung auf Linien Lösung auf dem Lösungsbereich anzeigen.

Lösung (alleine) Nur die Lösung anzeigen. Die Aufgabenstellung wird nicht gezeigt.

Aufgaben-Fußzeile Zustzinformationen nach der Aufgabe zeigen: Existierende Gruppenversionen, Gesamtpunktzahl.

Am Ende eines Dokuments können folgende Informationen gezeigt werden:

Notenliste Eine Liste mit Punkten und entsprechenden Noten.

Notenkasten Ein Bereich am Ende eines Dokuments, in dem Die erreichte Punktzahl und Note eingetragen wird.

Hier die Dokumenttyp-Argumente im Überblick:

3.1.1 Erläuterung der Argumente des Dokumenttyps

Von den folgenden Argumenten (siehe Tabelle 1, Seite 9) ist *nur eines* zu wählen, da sie sich gegenseitig ausschließen.

`[]` Ohne Argumente werden möglichst alle Teile der Aufgabe angezeigt.

`col` Mit dem Argument `col` wird eine Aufgabensammlung (Collection) ausgegeben, in der nur die Aufgaben sichtbar sind.

`lsg` Mit dem Argument `lsg` werden nur die Lösungen angezeigt. Somit kann ein Lösungsblatt erstellt werden. Da Dateinamen, Gruppeninformationen ebenfalls angezeigt werden, ist dieses Lösungblatt für Lehrer passend.

`slsg` Mit der Argument `slsg` (Schüler-Lösungen) werden nur die Lösungen angezeigt. Auf dem Lösungsblatt sind alle für Schüler wichtigen Informationen vorhanden.

`arb` Sollen keine Korrekturinformationen angezeigt werden,wird das Argument `arb` verwendet (Arbeitsblatt). Lösungslinien und Lösungskaros werden jedoch angezeigt.

`arblsg` wie `arb` jedoch mit Lösungen auf den Linien. So kann man ein ausgefülltes Arbeitsblatt erstellen. Dazu müssen bei der Lösungsangabe allerdings bestimmte Anforderungen erfüllt werden (siehe Seite 24**??**).

Tabelle 1: Dokumenttyp-Argumente der Dokumentklasse `teacher`

	Datei-/Versionsinfo	Aufgabenstellung	Lösungsbereiche	Korrekturhilfen	Gruppeninfo	Projektinfo	Lösung auf Linien	Lösung (alleine)	Aufgaben-Fußzeile	Notenliste	Notenkästen
ohne Argumente											
`[]`	x	x	x	x	x	x	x		x		
nur Aufgaben											
`col`		x									
nur Lösungen											
`lsg`	x				x	x	x	x			
`slsg`							x				
Arbeitsblätter											
`arb`		x	x								
`arblsg`	x	x	x				x				
Prüfungen											
`exam`		x	x								
`examlsg`	x	x	x				x				
Klassenarbeiten											
`ka`		x	x	x						x	x
`kalsg`	x	x	x	x			x			x	x
`kamulti`		x	x	x						x	x
`kamultilsg`	x	x	x	x			x			x	x

`exam` Für Prüfungen. Bislang identisch mit dem Argument `arb`.

`examlsg` Bislang identisch mit dem Argument `arblsg`.

`ka` Mit dem Argument `ka` werden die Aufgaben so formatiert, dass sie in Klassenarbeiten verwendet werden können. Es werden Lösungslinien erzeugt, erreichbare Punktzahlen angegeben und ein Punktzahlkasten erzeugt, in dem der korrigierende Lehrer die erreichte Punktzahl von Hand eintragen kann.

Am Ende Der Klassenarbeit wird ein Bereich ausgedruckt, in dem die Punkte zusammengezählt werden und die Noten angegeben werden können.

`kalsg` wie `ka` jedoch mit Lösungen auf den Linien.

`kamulti` Ebenfalls für Klassenarbeiten ist die Argument `kamulti` geeignet. Die Punkte werden getrennt nach Fächern aufsummiert. Somit können in *einer* Klassenarbeit Noten für *mehrere Fächer* vergeben werden. Um die Schüler zu informieren zu welchem Fach welche Teilaufgabe zählt, werden neben dem Punktzahlkasten ein Kürzel für das Fach angegeben (z.B. **M** für Mathe)

`kamultilsg` wie `kamulti` jedoch mit Lösungen auf den Linien.

3.1.2 Dokumenttyp im Dokument wechseln

Soll innerhalb eines Dokuments der Dokumenttyp gewechselt werden stehen folgende Befehle zur Verfügung:

Der Befehl `\kamulti` wechselt zum Dokumenttyp `kamulti`. Mit `\arblsg` wechselt man zum Dokumenttyp `arblsg`, ... usw.

Der Dokumenttyp kann nur *zwischen* 2 Aufgaben gewechselt werden. Deshalb müssen die Befehle auch *zwischen* 2 Aufgaben stehen.

3.2 Allgemeine Argumente

Folgende Argumenten können *zusätzlich* zum Dokumenttyp angegeben werden.

`sw` Will man auf einem Schwarz-Weiss Drucker ausdrucken, sollte man das Argument `sw` einschalten. Dann wird farbige Schrift Schwarz ausgedruckt.

Auf einzubindende Grafiken hat die Argument allerdings keinen Einfluss. Sie werden immer noch in ihren Originalfarben ausgedruckt. Mit einem Schwarzweissdrucker also in Graustufen.

`arial` Diese Argument stellt fast alle Schriftarten auf eine Arial-ähnliche Schrift um. Die Dokumente sehen dann in etwa so aus, als ob sie mit einem stinknormalem

WYSIWIG-Textverarbeitungsprogramm erzeugt wurden. Sinvoll nur für den, der so tun muss als würde er nicht mit LaTeX arbeiten, sondern mit W....

`debug` Mit dem Argument `debug` werden zusätzliche Informationen eingeblendet. So kann im formatierten Dokument z.B. erkannt werden, dass die Angabe des Schulnamens mit dem Befehl `\School{}` geschieht.

Außerdem werden bei Grafiken, die mit `\includepsfraggraphics` eingebunden werden, die Namen der Tags gezeigt und *nicht* ersetzt.

`frame` Wenn dies von dem verwendeten Seitenlayout unterstützt wird (Standardlayout, `bszleo`), kann mit dieser Argument ein Rahmen um das Blatt eingeblendet werden.

3.3 Argumente zum Erstellen von A 5-Blättern

`a5landscape` Mit dem Argument `a5landscape` wird ein DIN A 4 Blatt im Hochformat mit zwei *aufeinanderfolgenden* DIN A 5-Blättern Querformat (Engl. landscape) beschrieben.

`a5landscaperepeat` Für das Ausdrucken/Kopieren unter Streß ist es oft wünschenswert, dass z.B. auf Seite 1 des DIN A 4 Blatts *zweimal dasselbe* DIN A 5 Blatt zu sehen ist.

Dieses DIN A 4 Blatt braucht man nur auzudrucken, mittig zu halbieren und die beiden entstehenden Stapel aufeinanderzulegen, da sie ja aus den identischen DIN A 5 Blättern aufgebaut sind.

3.3.1 Bemerkungen zu den A 5-Argumente

Wenn sie die mit diesen Argumenten erzeugten `*.tex`-Datei mit dem Befehl `latex` bzw. `pdflatex` (`emacs`: Tastenkombination `shift-F5`) verarbeiten, dann bleibt allerdings die untere Hälfte des DIN A 4-Blattes leer.

Um das oben beschriebene Resultat zu erzielen, muss die `*.tex`-Datei mit `jefflatex` verarbeitet werden. Dies geschieht, wenn sie z.B. mit `emacs` die Taste F5 benutzen oder mit folgendem Befehl übersetzt wird:

```
jefflatex -f datei.tex
```

`jefflatex` ermitelt die zusätzlichen Argumente in der `*.tex`-Datei und stellt dann mit `pstops` die PostScript-Datei neu zusammen. Die direkte Erzeugung von `*.pdf` ist nicht möglich. Um `*.pdf` zu erhalten muss mit `ps2pdf` das mit `jefflatex` erzeugte PostScript nach `*.pdf` konvertiert werden.

3.4 Seitenlayout festlegen (Kopf- und Fußzeilen)

Ohne Angabe eines Arguments für das Seitenlayout ist das Seitenayout so festgelegt, dass wichtige Informationen (Schulname, Lehrerkürzel, Seitennummer, Dateiname, Versionsnummer und Ausdruckdatum) in der Kopfzeile bzw. der Fusszeile angezeigt werden können.

Wenn sie ein anderes Seitenlayout wählen wollen, stehen ihnen folgende vorgefertigte Seitenlayouts zur Verfügung:

`empty`
- Mit `empty` erhält man leere Kopf- oder Fußzeilen.

`simple`
- zeigt nur die Seitenzahl in der Fußzeile.(TODO ?????)

`grundschule`
- Mit dem Argument `grundschule` wird ein einfaches, schulneutrales Seitenlayout eingestellt, auf dem die Schüler oben ihren Namen eintragen können.

`bszleo`
- Das Argument `bszleo` erzeugt eine Formatierung, wie ich sie selbst am Beruflichen Schulzentrum Leonberg verwende.

  Wenn sie sehen wollen, wie bei mir ein Arbeitsblatt bzw. Klassenarbeit für die Schüler aussieht so geben sie `[bszleo,kamulti]` als Argumente an.

`bszleoexam`
- Mit `bszleoexam` wird ein Seitenlayout erzeugt, wie es in den Prüfungen für den Beruf des Kälteanlagenbauers am Beruflichen Schulzentrum Leonberg verwendet wird. Hier ist ein Deckblatt mit Inhaltsverzeichnis erforderlich, und auf den folgenden Seiten erscheint der Prüfungskopf.

  Wenn sie sehen wollen, wie an unserer Schule eine Prüfung aussieht, so geben sie `[bszleoexam,exam]` als Argumente an.

Soll ein Dokument von diesen mitgelieferten Seitenlayouts abweichen, so gibt es 2 Möglichkeiten:

1. Sie nutzen das Argument `empty`. Dann werden keinerlei Kopf- oder Fußzeilen ausgegeben. Mit ihrem Lieblings-WYSYWIG-Textverarbeitungs-Programm bedrucken sie dann diese Seiten nochmals, oder schnibbeln mit der Schere eigene Kopf- und Fußzeilen, die dann vor dem Gang zum Kopierer aufkleben.
2. Sie erstellen sich ein eigenes, an ihre Schule angepasstes Seitenlayout mit LaTeX.

Wenn sie die letztere Möglichkeit bevorzugen, also ein komfortables Arbeiten gewohnt sind, dann lesen Sie weiter.

3.4.1 Eigenes Seitenlayout erstellen

Wenn sie in den optionalen Argumenten der Dokumentklasse `teacher` ein nicht bekanntes Argument, wie zum Beispiel `myschool` angeben, dann sucht LaTeX nach folgenden Dateien:

- `myschool-aeoc.tex` — aeoc = At end of Class
- `myschool-abd.tex` — abd = At Begin Document
- `myschool-aed.tex` — aed = At End Document

Werden diese Dateien gefunden, so werden die darin enthaltenen Befehle, Zähleinstellungen, Kopfzeileneinstellungen, ... zum angegeben Zeitpunkt geladen.

Beispiel:

Sie wollen zu Beginn der Klassenarbeit Viel Glück wünschen. Dann schreiben sie in `myschool-abd.tex` die Worte:

```
Viel Glück bei der Klassenarbeit.
```

Die jeweils 3 Dateien der Seitenlayout-Argumente `bszleo`, `bszleoexam` und `grundschule` sind mit Kommentaren versehen, und können somit als Ausgangspunkt für eigene Seitenlayouts dienen.

3.4.2 Textfelder in eignen Seitenlayouts

Beim Erstellen von Vorlagen können die in Kapitel 4 angegebenen Felder benutzt werden. So kann z.B mit `\Quelle{Text}` eine Quellenangabe angegeben werden.

In der Vorlage kann dann an der gewünschten Stelle mit `\quelleuse{}` dieser Abgespeichere Text eingesetzt werden.

Ebenso kann auf alle anderen Texte zugegriffen werden.

3.5 Sonstige Argumente

`stoff` Für Stoffverteilungspläne gibt es das Argument `stoff`, das nur alleine verwendet werden sollte. Gemeinsam mit den vorigen Argumenten ergibt sich wenig Sinn.

Für die zur Verfügung gestellten Befehle siehe Seite 36

4 Ausfüllen des Kopfes im Seitenlayout

Je nach verwendetem Seitenlayout werden an vordefinierten Stellen der mit `teacher.cls` erstellten Dokumente Textfelder verwendet. Mit bestimmten Befehlen kann diesen Textfeldern eine Inhalt zugewiesen werden.

Für eine Übersicht der in einer Vorlage eingebauten Textfelder können sie das Argument `debug` mit angeben. Dann wird anstelle des Inhalts eines Textfelds der Befehl in rot ausgegeben, mit dem das Texfeld mit Inhalt gefüllt werden kann.

Im folgenden werden die Befehle zum füllen der Textfelder mit Inhalt beschrieben:

4.1 Befehle für Arbeitsblätter

`\School{}` Enthält den Schulnamen.

`\UserToken{}` Wird zum setzen des Lehrerkürzels benutzt.

`\Titelo{}` Gibt den Inhalt der oberen/ersten Titelzeile des Arbeitsblattes bzw. der Klassenarbeit an.

`\Titelu{}` Gibt den Inhalt der unteren/zweiten Titelzeile des Arbeitsblattes bzw. der Klassenarbeit an.

`\Fach{}` Damit wird ein Fach festgelegt, dem dieses Dokument zugeordnet werden kann.

`\Datum{}` Gibt das Datum an.

`\Quelle{}` Gibt die Quelle an. Sie steht je nach Seitenlayout an verschiedenen Stellen. Beim Arbeitsblatt mit Rahmen z. B. in der linken unteren Ecke des Rahmens.

`\Ausdruck{}` Gibt bei leerem Argument das Ausdruck-Datum und das Lehrerkürzel aus. Wird ein Argument angegeben, so wird dieses ausgegeben.

`\Revision{}` Gibt bei leerem Argument den Dateinamen und die CVS-Revisionsnummer/git-Datum an (falls vorhanden). Wird ein Argument angegeben, so wird dieses ausgegeben.

Für die Ausgabe der CVS-Versionsnummer/git-Datum von Aufgaben-Dateien wird folgende Zeile an den Anfang der Aufgaben-Datei gestellt (vor `\begin{aufgabe}`):

```
\documentclass{teacher}
\RCS $Revision$
```

Der Inhalt von `$Revision$` wird von CVS automatisch bei jeder Änderung gepached (verändert). Wie das mit git geht ist hier beschrieben: `http://www.linuxmuster.net/wiki/entw`

`\Entwurf{}` Markiert das ganze Dokument als Entwurf (Graue Schrift `Entwurf` im Hintergrund einer Seite). Außerdem wird die CVS-Revisionsnummer/git-Datum (falls vorhanden) und das Datum ausgegeben.

Wird ein Argument angegeben, so wird anstelle von `Entwurf` das Argument ausgegeben. Mit dem optionalen Argument kann die Größe dieser Hintergrundschrift angegeben werden.

Um den Text zu sehen, muss nach PostScript konvertiert werden (In `*.dvi` ist die Hintergrundschrift unsichtbar).

`\Klasse{}` Angabe der Klasse.

`\Blatt{}` Gibt die aktuelle Seitenzahl aus (leeres Argument). Wenn ein Argument angegeben wird, so wird dieses ausgegeben.

`\Name{}` Gibt den Schülernamen an.

4.2 Befehle für Klassenarbeiten

`\Fehler` Ändert das Wort `Fehler` im Notenkasten der Klassenarbeit um. Beispiel: `\Fehler{Punkte:}`

`\Muendlich` Wie bei `\Fehler`

4.3 Prüfungsdauer (nur bei `bszleoexam`)

Folgende Optionen sind bei `bszleoexam` zwingend erforderlich:

`\APtime` Prüfungsdauer im Fach Arbeitsplanung.

`\Ttime` Prüfungsdauer im Fach Technologie.

`\Mtime` Prüfungsdauer im Fach Mathematik.

`\Totaltime` Gesamt-Prüfungsdauer.

4.4 Benutzerdefinierte Einstellungen

Jeder Bearbeitungsvorgang sucht im Homeverzeichnis des aufrufenden Users nach der Datei `.problectix/problectix.tex` und führt die darin enthaltenen LaTeX-Befehle aus.

Steht in dieser Datei z. B. `\UserToken{jeffbeck}`, erscheint an den entsprechenden Stellen der Vorlage `jeffbeck` (Wenn die Einstellung nicht nach `\begin{document}` im zu bearbeitenden Dokument überschrieben wird).

5 Lizenzinformationen

Um Creative Commons Lizenzinformationen anzugeben, wird das Paket `cclicenses` benutzt.

Folgende Lizenzen 6 sind nach Version 2.0 möglich:

LaTeX-Befehl	Bedeutung	Symbol
`\by`	Namensnennung	
`\bynd`	Namensnennung Keine Bearbeitung	
`\byncnd`	Namensnennung Nicht Kommerziell Keine Bearbeitung	
`\bync`	Namensnennung Nicht Kommerziell	
`\byncsa`	Namensnennung Nicht Kommerziell Weitergabe unter gleichen Bedingungen	
`\bysa`	Namensnennung Weitergabe unter gleichen Bedingungen	

6 Erstellen von Aufgaben

6.1 Dateinamen — Konventionen

Jede Aufgabe wird in einer Datei abgelegt. Diese Datei wird in ein passendes Verzeichnis abgelegt, um die Übersichtlichkeit zu bewahren.

Z. B. in `mytex/aufgaben-kb/3-lehrjahr/Verdichter/`

Konventionen für die Dateinamen:

- Dateinamen beginnen mit Großbuchstaben.
- Die Dateinamen sollten aussagekräftig sein, und ein Gebiet umfassen (Verdichter, Regelkreis, Verflüssiger, ...)
- Nach dem Dateinamen sollte das Fach in Kurzform vermerkt sein, dem die *gesamte* Aufgabe zugeordnet ist.
- Die einzelnen Aufgaben eines Gebietes werden mit einer dreistelligen Nummer versehen.

- Aufgaben, die in leicht veränderter Form in einer Prüfung verwendet wurden, sind mit dem Zusatz `-pr` versehen.
- Beinhaltet eine Aufgabe ein `*.eps`-Datei so ist derselbe Name zu wählen.

Beispiele für Dateinamen:

- `Verdichter-m-001.tex`

  mit den Grafiken

  - `Verdichter-m-001a.eps`
  - `Verdichter-m-001b.eps`
- `Regelkreis-t-002.tex` mit `Regelkreis-t-002.eps`
- `Regelkreis-m-002-pr.tex`

6.2 Erstellen der Aufgabenstellung

Jede Aufgabe wird in einer Datei abgespeichert. Diese Datei muss folgendermaßen aufgebaut sein:

```
\begin{aufgabe}[Fach]{Aufgabentitel}
   \begin{textonly}                          %% optional
      ... %% Text vor den Teilaufgaben       %% optional
   \begin{textonly}                          %% optional
   \begin{teilaufgabe}[o]{Teilaufgabenfach}{Linienzahl}{Punkte}
       \kariert                              %% optional
       Teilaufgabentext
       ...
       \korrektur{Länge}...                  %% optional
   \end{teilaufgabe}
   \begin{loesung}                           %% optional
       \punkte{Richtige Antwort zu 1}        %% optional
              {Punkteanzahl zu 1}            %% optional
              {Kommentar zu 1}               %% optional
       \punkte{Richtige Antwort zu 1}        %% optional
              {Punkteanzahl zu 2}            %% optional
              {Kommentar zu 2}               %% optional
   \end{loesung}                             %% optional
\end{aufgabe}
```

Die verwendeten Umgebungen und ihre Argumente werden im folgenden erläutert.

`aufgabe` Die Umgebung `aufgabe` beinhaltet die *gesamte* Aufgabe. Sie erstellt Kopf- und Fußzeile. Das optionale Argument `Fach` ordnet der *gesamten* Aufgabe ein Fach zu. Dieses Fach hat im weiteren keine Bedeutung mehr. Bei Fächerübergreifenden Aufgaben dient `Fach` als Orientierung für den Schwerpunkt der Aufgabe.

Das Argument `Aufgabentitel` gibt jeder Aufgabe eine Überschrift, die deren Inhalt kurz umreißt. Sie sollte knapp gewählt werden, da kein Zeilenumbruch vorgesehen ist (z.B. `Verdichterberechnung`).

`teilaufgabe` Die Umgebung `teilaufgabe` umschließt die Aufgabenstellung einer Teilaufgabe. Das optionale Argument `ohnenummer` bzw. `o` sorgt dafür, dass Teilaufgaben keine Nummerierung erhalten. Dies ist notwendig, wenn die Aufgabe nur eine Teilaufgabe enthält.

Das Argument `Teilaufgabenfach` ordnet jede Teilaufgabe einem Fach zu. Mögliche Werte für `Teilaufgabenfach` sind `T`, `t`, `AP`, `ap`, `Ap`, `M`, `m`

Je nach Art des Faches werden karierte (`M`) oder linierte Lösungsbereiche (`T`, `AP`) erzeugt.

`kariert` `liniert` Mit `\kariert` bzw. `\liniert` zu Beginn der Umgebung `teilaufgabe` die Linienart verändert:

`singlepzk` `doublepzk` Mit `\singlepzk` bzw. `\doublepzk` können sie zwischen einem einfachen oder doppelten Punktzahlkasten wählen. Standart ist der einfache Kasten.

Durch die Angabe von `Teilaufgabenfach` können bei Klassenarbeiten die mehrere Fächer beinhalten (Fächerverbindenden Klassenarbeiten) Punkte nach Fächern getrennt aufaddiert werden, wenn eines der Argumente `arblsg`, `examlsg`, `kalsg` oder `kamultilsg` (Siehe Seite 9) verwendet wird.

Das Argument `Linienzahl` gibt an wieviele Lösungslinien erscheinen sollen. Wurde als Fach Mathe angegeben, wird statt den Lösungslinien ein kariertes Feld ausgegeben. Das Argument `Linienzahl` gibt dann die Höhe des Felden in Kästchen an.

Ist die `Linienzahl` null, dann erscheint ein Punktzahlkasten ohne Lösungslinien.

Das Argument `Punkte` gibt an, wieviele Punkte in dieser Teilaufgabe erreicht werden können. Diese Punktzahl erscheint neben dem Punktzahlkasten.

6.2.1 Normaler Text in einer Aufgabe

`textonly` Mit der Umgebung `textonly` wird ein Textblock innerhalb einer Aufgabe erzeugt. Der Textblock hat keine Nummerierung, keine Maximalpunktzahl und keinerlei Lösungslinien. Intern werdern die Textblöcke einer Aufgabe jedoch mit negativen Zahlen durchnummeriert. Mit `\ohne[-1][2]{-4}` werden der erste Textblock(-1),

die zweite Teilaufgabe (2) und der vierte Textblock (-4) verborgen.

Zu beachten ist, dass jeglicher Text der Aufgabenstellung entweder in der `\textonly`-Umgebung oder in der `teilaufgabe`-Umgebung steht. Nur so kann bei einem Lösungsblatt die Aufgabenstellung vollständig ausgeblendet werden.

6.2.2 Multiple-Choice-Aufgaben (Ankreuzfragen)

`mch` Die Umgebung `mch[x]` erzeugt eine Aufzählung mit dem Ankreuzkästchen ☐ als Aufzählungszeichen. Das Zusatzargument x (Zahl ohne Längenangabe) rückt die Liste um $x$mm ein.

Die Ankreuzmöglichkeiten werden mit `\item` realisiert. Bei korrekten Antworten sollte `\itemx` angegeben werden, damit wird das Kästchen angekreuzt, wenn eines der Argumente `arblsg`, `examlsg`, `kalsg` oder `kamultilsg` verwendet wird (Siehe Seite 9). Das optionale Argument `\itemx[A]` setzt statt dem Kreuz das optionale Argument in das Kästchen (hier A). So können Ankreuzfragen z.B. durchnummeriert oder A oder B zugeordnet werden.

Wird `mch` in die Umgebung `multicols{n}` geschachtelt, wird die Multiple-Choice-AufgabenAufzählung auf `n` Spalten verteilt. Dazu wird von `teacher.cls` das Package `multicol` dazugeladen.

Die Verschachtelungsreihenfolge für mehrspaltige Multiple-Choice-Aufgaben ist:

```
\begin{teilaufgabe}
   %% Beginn der mehrspaltigen Multiple-choice-Aufgabe
   \begin{multicols}{n} %% Falls mehr als eine Spalte ...
      \begin{mch}[2]
         \item  ... Ankreuzmöglichkeit 1 ...
         \item  ... Ankreuzmöglichkeit 2 ...
         \itemx ... Ankreuzmöglichkeit 3 (korrekt) ...
         \item  ... Ankreuzmöglichkeit 4 ...
            ...
         \end{mch}
   \end{multicols}
   %% Ende der mehrspaltigen Multiple-choice-Aufgabe
\end{teilaufgabe}
```

Sollen in Ankreuzaufgaben A-Version und B-Version genutzt werden, geht die nur innerhalb der eckigen Klammer von `\itemx` sowie dem entsprechenden Text:

```
\begin{mch}[2]
   \itemx[\ab{---}{X}] \ab{Lüge}{Wahrheit}
   \itemx[\ab{X}{---}] \ab{Richtig}{Falsch}
```

```
   ...
\end{mch}
```

Um all Einrückungen gleich zu halten, sollte jedesmal `\itemx[\ab{X}{---}]` verendet werden.

6.3 Einbinden von Grafiken

6.3.1 Grafik und Text nebeneinamder

`linksbild` Mit der Umgebung

```
\begin{linksbild}[Bildtitel]{*.eps-Datei}{Bildbreite in mm}
   ... Text
\end{linksbild}
```

oder

```
\begin{xlinksbild}[Bildtitel]{*.eps-auf}{*.eps-lsg}{Bildbreite in mm}
   ... Text
\end{xlinksbild}
```

können Bilder an den linken Rand mit Text rechts daneben erstellt werden. Das Bild und ebenso der Text befinden sich jeweils in einer Minipage. Fussnoten im `Text` erscheinen direkt unterhalb des `Textes`.

Beachten Sie , dass das Bild nicht vom Text umflossen wird.

??? neue Umgebung mit umflossenenem Bild ...

Beide Minipages sind in einer Minipage mit voller Breite (genauer: `textwidth`) zusammengefasst. Deshalb werden Sie immer gemeinsam umbrochen.

Folgende Argumente sind in `linksbild` möglich:

`Bildtitel` gibt den Text an, der unter dem Bild stehen soll. Er ist unter dem Bild zentriert. Wenn er länger als die Bildbreite ist, wird er umbrochen (zentriert).

Wenn sie die Bilder Nummerieren wollen, dann können sie `Bildtitel` und den Text, der auf dieses Bild verweist so wählen:

`Bild \theaufgabennummer .1`

??? In Zukunft sollen Bilder automatisch in jeder teilaufgabe durchnummeriert werden. Dann würde der Text lauten:

`Bild \theaufgabennummer .\thebildnummer`

`*.eps-Datei` gibt den Namen der `*.eps`-Datei an, die eingefügt wird. Die Endung `.eps` ist wegzulassen.

`Bildbreite` ist ein Zahlenwert ohne Einheit und gibt die Breite des Bildes (in mm) auf der linken Seite an. In den Bereich rechts des Bildes wird der `Text` geschrieben, der innerhalb der `linksbild`-Umgebung steht.

Der `Abstand` zwischen Bild und Text kann mit folgender Längenzuweisung eingestellt werden:

```
\setlength{\bildtextsep}{Abstand}
```

Standardmäßig wird ein Abstand von 8 mm benutzt. Dieser Wert kann *vor* der `linksbild`-Umgebung verändert werden und bleibt bis zu nächsten Änderung erhalten.

Die Bilder können tiefergestellt werden mit

```
\setlength{\bildtiefer}{Tieferstellung}
```

Standard für die Tieferstellung ist `-1.25ex` (Wieso nicht 0???).

6.3.2 Abbildungen über die ganze Seitenbreite

`includegraphics` Mit `\includegraphics` kann man Bilder aus `*.eps`-Dateien einbinden:

```
\begin{center}
   \includegraphics[width=145m]{Datei}
\end{center}
```

Dies bindet Die Datei `Datei.eps` mit einer Breite von 145 mm ein.

Wenn die `*.eps`-Dateien mit einem geeigneten Vektorgrafik-Programm, z.B. `xfig` (Siehe Seite 51) erstellt wurden, kann man den in den Grafiken enthaltenen Text ersetzten. Dazu ist das Paket `psfrag` erforderlich, das von der Dokumentklasse `teacher` automatisch geladen wird.

`includepsfraggraphics` Wenn man `*.eps`-Dateien einbindet, in denen Ersetzungen mit `psfrag` vorkommen, sollte man diese Ersetzungen *nicht* im Dokument vornehmen, da die Grafik dann nur durch Kopie dieser Ersetzungen in anderen Dokumenten wiederverwendbar ist.

Besser ist es, zur Datei `grafikname.eps` eine Datei `grafikname.tex` anzulegen und die Grafik mit `\includepsfraggraphics{grafikname}` einzubinden. Dieses

Vorgehen lädt automatisch die Ersetzungen aus der Datei `grafikname.tex` vor der Grafik. Die erstellte Grafik kann dann mit `\includepsfraggraphics{grafikname}` wiederverwendet werden.

6.3.3 Ganzseitige Grafiken mit festem Maßstab

Oft werden ganzseitige Grafiken erforderlich, die ohne Vergrößerungsfaktor eingebunden werden sollen (Beispiel: Isometriepapier).

Dies geschieht mit den Befehlen:

```
\newarbchecked{Kommentar}{leer-a4.epsi}
```

welcher einen neuen Arbeitsblattkopf erzeugt, bzw.:

```
\newpagechecked{Kommentar}{leer-a4.epsi}
```

welcher ein neues Blatt nur mit Kopfzeile erzeugt.

wobei `Kommentar` als `\Titelu{Kommentar}` verwendet wird. Mit dem Optionalen Argument kann man eine Datei angeben, die verwendet wird. Möglich Dateinamen (Grafiken sind schon in `problectix` enthalten) sind:

```
kariert-a4.epsi
isometrie-a4.epsi
leer-a4.epsi
```

Wenn man eigene Grafiken erstellen möchte ist es sinvoll Die Datei `leer-a4.dxf` mit `librecad` zu bearbeiten. Damit ist sichergestellt, dass man innerhalb des 175mm x 250mm großen Rahmens eine Grafik erstellen kann, die ein Blatt DIN A 4 ausfüllt.

Man druckt dann die Datei mit `librecad` aus (`*.ps`-Datei) und erzeugt dann mit dem Befehl `ps2epsi datei.ps` eine `*.epsi`-Datei, die mit den obigen Befehlen eingefügt wird.

Die im nächsten Kapitel dargestellten Anhänge sind recht ähnlich.

6.3.4 Ganzseitige Grafiken im Anhang

Bei vielen Aufgaben sind Grafiken erforderlich, die über eine ganze Seite gehen. (Für kleine Grafiken siehe Seite 21.)

`anhang` Diese ganzseitigen Grafiken können als Anhang an das eigentliche Dokument (Arbeitsblatt, Klassenarbeit, Prüfung) angehängt werden. An der Stelle in einer Teilaufgabe an der auf die Grafik im Anhang verwiesen werden soll, steht der Befehl:

```
\anhang[graphicx-argument]{Beschreibung}{Datei.eps}
```

bzw. wenn für Aufgabenstellung und Lösung 2 verschiedene Dateien angehängt werden sollen:

```
\xanhang[graphicx-arg.]{Beschreibung}{Aufgabe.eps}{Lsg.eps}
```

Das Argment `Datei.eps` (bzw. `Aufgabe.eps` und `Lsg.eps`) gibt an, in welcher Datei sich die anzuhängende `*.eps`-Grafik befindet.

Mit der `Beschreibung` wird der Inhalt der Grafik näher beschrieben. Die `Beschreibung` findet man auf der Anhangseite unten links in der Fusszeile wieder. Benutzt man ein Seitenlayout, das eine Inhaltsangabe erzeugt (z.B. `bszleoexam`), so erscheint die Anhangseite mit ihrer Seitenzahl im Inhaltsverzeichnis.

Die erste Angangseite bekommt den Titel **Anhang A**, die zweite Anhangseite **Anhang B**, usw.

Mit dem optionalen Argument `graphicx-argument` können an das `graphicx`-Paket Argumente übergeben werden. Ohne diese Angaben, werden die Grafiken im Anhang auf die maximale Größe skaliert, wobei der Vergrößerungsfaktor in x und y-Richtung konstant ist.

Überlegenswert als `graphicx-argument` ist z.B. `width=120mm`. Damit wird die Breite der Grafik auf 120mm festgelegt. Für die kompletten Möglichkeiten lesen sie bitte die Dokumentation zum Paket `graphicx.sty`.

6.4 Abgleichen des Aufgabenendes

In seltenen Fällen ist ein manueller Abgleich des Aufgabenendes erforderlich.

Wird in einer Aufgabe ohne abschließende Lösungslinien am Ende der Aufgabenstellung eine Grafik verwendet, dann sollte der Rand der Grafik mit der Unterkante des Punktzahlkastens fluchten. Besitzt die Grafik einen weißen Rand, so entsteht ein *flatterndes Teilaufgabenende*.

Um Platz zu sparen und zwischen den Teilaufgaben bzw. Aufgaben einen einheitlich breiten *whitespace* zu erhalten sollten diese flatternden Aufgabenenden *manuell* abgeglichen werden.

Dieser Abgleich geschieht folgendermaßen:

1. In der Dokumentklasse `teacher` wird das Argument `debug` genutzt. Diese Option sorgt unter anderem dafür, dass an der Unterkante eines jeden Punktzahlkastens eine schmale, horizontale Linie erscheint. Diese hat im linken Randbereich die rote Kennzeichnung `DEBUG`, und 2 Abstandsmessskalen mit Linien im Millimeterabstand wie auf zwei senkrecht angelegten Linealen.

2. Im erzeugten Postscript-File kann nun abgelesen werden, um wieviele mm der Aufgabenkasten nach oben (Regelfall) oder nach unten verschoben werden muss, um mit der Aufgabenunterkante zu fluchten.

3. Mit dem Befehl `\korrektur{längenangabe}` wird die Korrektur als letztes vor dem Beenden der `teilaufgabe`-Umgebung (`\end{teilaufgabe}`)eingetragen. Der Befehl `\korrektur{längenangabe}` ruft lediglich den Befehl `\vspace{längenangabe}` auf. Ein neuer Befehl wurde deshalb definiert um *nachträglich* in jeder Aufgabe noch Befehle nachtragen zu können, falls dies erforderlich wird.

6.5 Erstellen der Lösungen

6.5.1 Endergebnisse/Kurzlösungen im Lösungsbereich

Lösungen erscheinen im Ausdruck bei Verwendung von `arblsg`, `examlsg`, `kalsg` oder `kamultilsg` (Siehe Seite 9) in den Lösungsbereichen in grüner Schrift.

Bei Verwendung von `lsg` oder `slsg` erscheinen Sie schwarz (Nur Lösungen).

`loesung` Innerhalb der Umgebung `loesung` wird die Lösung eingegeben. Sie zählt zur vorherigen Teilaufgabe. Zum Aufbau einer Aufgabe siehe Seite 17.

`punkte` Der Befehl `punkte` darf nur innerhalb der `loesung`-Umgebung verwendet werden und hat 3 Argumente:

```
\punkte{Richtige Antwort}{Punkte}{Kommentar}
```

Das Argument `Richtige Antwort` ist der Lösungsvorschlag.

Das Argument `Punkte` gibt die Punktzahl an, die für `Richtige Antwort` vergeben wird. Diese Punktzahl wird *nicht* weiterverwendet für ausfummierungen usw.

Das Argument `Kommentar` hat die Aufgabe für den Lehrer beim korrigieren zusätzliche Informationen zur Verfügung zu stellen. Z.B. ... zählt nur einen halben Punkt ....

Wenn Linien als Lösungsbereiche verwendet werden, beschreibt *ein* `punkte`-Befehl *eine* Lösungslinie. Am Ende des Befehls folgt ein Zeilenumbruch.

Werden Karos als Lösungsbereiche verwendet, dann werden die `Richtige Antwort`en der `punkte`-Befehle *ohne* Zeilenumbruch aneinandergehängt. Soll ein Zeilenumbruch erscheinen, muss am Ende des `punkte`-Befehls LaTeX-Umbruchbefehl `\par` stehen:

```
\punkte{Richtige Antwort}{2}{}\par
```

Wenn `\par` bei einer Aufgabe mit Linien steht, dann werden 2 Zeilenumbrüche eingefügt (Das ist nicht das was sie wünschen). Der zweite Zeilenumbruch wird in Zufunft wegfallen(????????).

Standardmäßig wird beim Erstellen der Lösung nach einem Punktebefehl *keine* neue Zeile begonnen. Ist dies erwünscht, dann muss nach einem `\punkte{}{}{}`-Befehl der LaTeX-Befehl `\par` stehen.

6.5.2 Musterlösungen/ausführliche Lösungen

Bei der Verwendung von Karos als Lösungsbereich kann zusätzlich zum (kurzen) Endergebnis eine ausführliche Musterlösung in der farbe lila angegeben werden.

Das grüne Endergebnis füllt den Lösungsbereich von unten her auf, die lila Müsterlösung (der Lösungsweg) füllt von oben her auf.

6.5.3 Lösungen an beliebigen Stellen in der Aufgabenstellung

Hat man in der Aufgabenstellung eine Fotografie verwendet, in der als Lösung etwas eingezeichnet werden soll, kann dies mit:

```
\punkte[x-Wert, Y-Wert]{Hier}{2}{}
```

erfolgen. Als `X-Wert` bzw. `Y-Wert` werden Zahlen ohne Einheit angegeben. Diese Werte geben den Abstand vom linken, unteren Aufgabenende an. Mit etwas rumprobieren kann man die Lösung `Hier` auf der Fotografie positionieren.(Wenn `\renewcommand{\baselinestretch}{x}` verwendet wird, dann können sich diese Positionen verschieben. (Sollte in zukünftigen Versionen nicht mehr so sein ??????).

Benutzt man eine Vektorgrafik, ist es besser 2 `*.eps`-Dateien zu erzeugen. Eine für die Aufgabenstellung und eine für die Lösung. Die Grafiken sollten dieselbe Größe haben, damit sich der Zeilenumbruch nicht verschiebt. Mit folgendem Konstrukt werden Aufgabengrafik oder Lösungsgrafik Abhängig vom Zähler `xlsg` augegeben:

```
\xincludegraphics[Option]{Datei}{Datei-lsg}
```

oder bei Dateien mit `psfrag`-Ersetzungen (Siehe auch 6.3.2):

```
\xincludegraphics[Option]{Datei}{Datei-lsg}
```

Noch zu testen.??????

6.5.4 Lösungen mit großem Umfang

Es gibt Lösungen die deutlich umfangreicher sind als ihre Aufgabe. Beispiel: Zeichnen sie auf ein Blatt folgendes Werkstück in 3 Ansichten und bemaßen sie es.

Diese Lösungen können, wenn die Lösung gezeigt werden soll, an das Arbeitsblatt angehängt werden.

Alles, was in der Option des folgenden Befehls geschrieben wird, wird nur gezeigt, wenn die Lösungen gezeigt werden:

```
\xlsgonly{
   \newpage

   Lösung zum ersten Bild:

   \includegraphics{file-1.epsi}

   \newpage

   Lösung zum zweiten Bild:

   \includegraphics{file-2.epsi}
}
```

6.5.5 Lösungen bei Multiple-Choice-Aufgaben

Siehe Seite 19.

6.5.6 Lösungen bei der `linksbild`-Umgebung

Siehe Seite 20.

6.5.7 Lösungen bei Anhängen

Siehe Seite 22.

7 Erstellen einer Klassenarbeit aus Aufgaben

7.1 Klassenarbeits-Datei

Die Grundstruktur einer Klassenarbeits-Datei sieht folgendermaßen aus:

```
\documentclass[11pt,ka]{teacher}
   \Titelu{Klassenarbeitstitel}
   \School{Klassenarbeitstitel}
   \UserToken{Klassenarbeitstitel}
   \Fach{m}
   \gruppea
\begin{document}
   \nehme{aufgabe-1}
   \nehme{aufgabe-2}
   \nehme{aufgabe-3}
   \ohne[-1][2]{1}
   \nehme{aufgabe-4}
\end{document}
```

In der oberen Titelzeile steht bei Verwendung von `ka`, `kalsg`, `kamulti` bzw. `kamultilsg` (Siehe Seite 9) die Angabe „Klassenarbeit“. Dieser Inhalt kann mit dem Befehl `\Titelo{Text}` durch `Text` ersetzt werden. Die untere Titelzeile kann mit `\Titelu{text}` angegeben werden.

Für weitere sinnvolle Angaben siehe Seite 14 ff.

7.2 Einfügen von Aufgaben

`nehme` Um eine in einer Datei abgespeicherte Aufgabe in eine Klassenarbeit (Prüfung, Arbeitsblatt, ...) einzubinden wird der Befehl

```
\nehme[option]{Dateiname}
```

verwendet.

Um den LaTeX-Quellcode der Datei `Dateiname` einzufügen, wird anstelle von `option` der Wert `quellcode` angegeben.

Mit der Option `[beispiel]` wird eine Aufgabe ausgegeben *und* deren Quellcode angefügt.

`nehmealle` Bei Aufgaben, innerhalb derer mit `\ab{...}` - und `\abc{...}` - Befehlen verschiedene Gruppen definiert wurden, können *alle* Gruppen nacheinander angezeigt werden, wenn sie mit dem Befehl

```
\nehmealle{Dateiname}
```

Die Stellen, an denen sich die A, B und C-Version unterscheiden sind farblich hervorgehoben

Seit Juli 2009 gibt es `\abcd{...}` bis `\abcdefgh{...}`. Wie der Befehl reagiert ist ungewiss.

`allealle` Für eine *Aufgabensammlung* ist im Vorspann der Befehl

```
\allealle
```

sehr nützlich. Es definiert den Befehl `\nehme{Dateiname}` in den Befehl `\nehmealle{Dateiname}` um. So kann von *nur eine Gruppe anzeigen* (Standardeinstellung) auf *alle erzeugten Gruppen aller Aufgaben zeigen* umgestellt werden. Der Umfang und die Rechenzeit für die Aufgabensammlung kann sich dabei natürlich vervielfachen (max. 3 x).

7.3 Modifizieren schon vorhandener Aufgaben

Wenn man Aufgaben aus einer Datenbank benutzt, kann man sie mit folgenden Befehlen modifizieren.

`ohne` Das Unterdrücken von Teilaufgaben erfolgt mit dem Befehl:

```
\ohne[Teilaufgabe][Teilaufgabe]{Teilaufgabe}
```

Als Argument `Teilaufgabe` können positive Werte stehen. Dann wird die entsprechende Teilaufgabe weggelassen.

Verwendet man als Argument für `Teilaufgabe` einen negativen Wert, werden die durch die Umgebung `textonly` erzeugten Textblöcke weggelassen.

Der Befehl `\ohne` kann nur *einmal* vor jeder Aufgabe verwendet werden.

8 Klassenarbeiten mit mehreren Gruppen bzw. Projektbezug

8.1 Unterschiedliche Aufgabenvarianten (A, B, C)

`ab` Um aus einer Aufgabe mehrere Varianten zu erzeugen, gibt es die beiden Befehle

```
\ab{Text/Befehle der Gruppe A}{Text/Befehle der Gruppe B}
```

`abc` und

```
\abc{Text/Befehle von A}{Text/Befehle von B}{Text/Befehle von C}
```

Die Befehle werden verwendet, wenn 2 bzw. 3 alternative Fragestellungen erzeugt werden sollen.

Sie dürfen *nicht beide* in einer Aufgabe vorkommen, da dies zweideutig wäre. Natürlich darf z. B. in der Klammer von `Text/Befehle von A` und `Text/Befehle von C` dasselbe stehen.

In einem Dokument ist standardmäßig `\gruppec` eingestellt, und es erscheint *keine* Gruppenangabe im KA-Rahmen.

Bei einer Aufgabe ohne die Befehle `\ab{...}` oder `\abc{...}` wird die Aufgabe immer unmodifiziert ausgegeben.

Bei einer Aufgabe mit dem Befehl `\ab{...}` ist keine Gruppe C vorhanden. Stattdessen wird der Eintrag von Gruppe A benutzt.

`gruppea`
`gruppeb`
`gruppec`

Wird die Gruppenauswahl mit einem der folgenden Befehle umgestellt werden:

`\gruppea`, `\gruppeb` oder `\gruppec`

so wird bei allen Aufgaben, in denen die Befehle `\ab-` oder `\abc` verwendet wurden, die zu der entsprechenden Gruppe gehörenden Texte eingesetzt.

Wird schon im Vorspann einer der Befehle verwendet so ändert sich Der obere/erste Titel zu „Klassenarbeit (A)“

Die Befehle `\ab` und `\abc` können prinzipiell auch dazu verwendet werden um *gänzlich* verschiedene Aufgaben alternativ zu verwenden. Man sollte jedoch bedenken, dass aufgrund des unterschiedlichen Platzbedarfs dieser Aufgaben die Klassenarbeiten unterschiedlich lang ausfallen können und ein unterschiedlicher Seitenumbruch in Version A,B und C entstehen kann.

8.1.1 Hinweis

Werden verschidene Lösungen für A,B,C angegeben, so müssen die Lösungen inerhalb der `\punkte{}{}{}` angegeben werden, also so:

```
\punkte{\abc{Berlin}{Paris}{Rom}}{1}{Hinweis}
```

8.2 Unterschiedliche Aufgabenvarianten (D, E, F, G, H)

Seit Juli 2009 gibt es die weiteren Befehle

```
\abcd{}{}{}{}
\abcde{}{}{}{}{}
```

```
\abcdef{}{}{}{}{}{}
\abcdefg{}{}{}{}{}{}{}
\abcdefgh{}{}{}{}{}{}{}{}
```

sowie

```
\grupped
\gruppee
\gruppef
\gruppeg
\gruppeh
```

Sie erweitern die bisherigen Befehle um bis zu 8 verschiedene Aufgabenvarianten zu erzeugen.

8.3 Projektaufgaben

`projekt` In *einer* Aufgabe kann eine normale (=nicht projektbezogene), sowie eine projektbezogene Fragestellung erzeugt werden. Dies geschieht mit dem Befehl:

```
\projekt{projektbezogen}{nicht projektbezogen}
```

`projektbezug` Standardmäßig wird der Inhalt von `{nicht projektbezogen}` in der Klassenarbeit erscheinen. Wenn jedoch im Vorspann der Klassenarbeit der Befehl

```
\projektbezug
```

steht, wird anstelle des Inhalts von `{nicht projektbezogen}` der Inhalt von `{projektbezogen}` eingesetzt.

Dies ermöglicht, schon vorhandene Aufgaben nachträglich mit einer projektbezogenen Fragestellung zu versehen.

Beispiel:

```
In \projekt{der beschriebenen}{einer} Kälteanlage wird ...
```

9 Umfangreiche Dokumente (Prüfungen)

Bei umfangreichen Dokumenten ist es sinnvoll eine Untergliederung z.B nach Fächern vorzunehmen.

`examfach` Um eine Gliedeungs-Überschrift zu erzeugen dient der Befehl

```
\examfach[Marke]{FachTitel}
```

mit dem in einer Schattenbox der Text `FachTitel` angezeigt wird. Es wird jedoch keine neue Seite begonnen.

Mit dem optionalen Argument `Marke` wird die Aufgabenmarke gesetzt. Die Aufgabenmarke ist der Nummer vorangesetzt, z.B. ist bei der Nummerierung T1, T2, T3, ... T die Aufgabenmarke.

Gleichzeitig wird ein Eintrag ins Inhaltsverzeichnis erstellt. Ein Inhaltsverzeichnis erzeugt automatisch z.B. das Seitenlayout `bszleoexam` auf der ersten Seite.

`ExamType` Um verschiedene Typen von Prüfungen zu erzeugen wird

```
\ExamType{number}
```

verwendet. Beim Seitenlayout `bszleoexam` bedeutet z.B. `\ExamType{1}` „projektunabhängig“ und `\ExamType{2}` „projektabhängig“.

10 Erstellen von Arbeitsblättern

10.1 Einführung

Arbeitsblätter sind in ihrem Seitenlayout kaum standardisierbar. Deshalb können hier nur Anregungen gegeben werden.

10.2 Gliederungsbefehle

10.2.1 Mehrere Arbeitsblätter in einer Datei

Die Dokumentklasse `teacher` greift *nur am Beginn eines Dokuments* auf eigene Seitenlayouts zurück. Das mag in Ordnung sein, wenn z.B. eine 5-seitige Klassenarbeit gedruckt und zusammengetackert wird. Hier reicht das einmalige Einlesen des Seitenlayouts (Kopf mit Namensfeld, ...) am Dokumentbeginn aus.

Möchte man jedoch aufeinander aufbauende Arbeitsblätter erstellen, dann ist es oft notwendig, unabhängige Arbeitsblätter in einer Datei zusammenzufassen. Das Seitenlayout der ersten Seite sollte mehrmals erscheinen.

`newarb` Ein neues Arbeitsblatt kann mit dem Befehl

```
\newarb{Unterer Titel}
```

begonnen werden. Dann wird eine neue Seite mit dem Seitenlayout der ersten Seite begonnen. Mit `Unterer Titel` wird der Befehl `\Titelu{Unterer Titel}` ausgeführt, der üblicherweise dem neuen Arbeitsblatt einen neuen Titel gibt. Der

obere Titel bleibt durch das ganze Dokument durchgehend gleich (falls man ihn nicht mit `\Titelo` ändert).

10.2.2 Überschriften

`arbsection` `arbsubsection` `arbsubsubsection`

Die von LaTeX verwendeten Gliederungsbefehle wie z.B. `\section` sind für Arbeitsblätter zu raumfordernd. Deshalb stehen für Überschriften die Befehle

```
\arbsection{Titel}
\arbsubsection{Titel}
\arbsubsubsection{Titel}
```

zur Verfügung. Sie besitzen keine Nummerierung.

10.2.3 Versuche

`versuch`

Um Versuche vom restlichen Text abzuheben wird der Befehl

```
\versuch{Titel}
```

benutzt. Der `Titel` wird von einem Kästchen umrahmt und mit einer nummerierten Versuchsnummer versehen, die im Dokument automatisch durchgezählt wird.

Soll der Zähler der Versuche auf einen bestimmten Wert `n` gesetzt werden, geschieht dies mit:

```
\setcounter{versuch}{n}
```

10.3 Leerbereiche in die Schüler etwas eintragen sollen

10.3.1 Grundlegendes

Damit ein Schüler handschriftlich ein einzelnes Wort oder zusammenhängenden Text eintragen kann, muss der Zeilenabstand erhöht werden. Die geschieht mit dem Befehl `\lue` und gilt bis zum Ende der laufenden Umgebung oder des laufenden Absatzes (ODER ????). Im nachfolgenden finden sich Beispiele.

10.3.2 Lücken innnerhalb von Fließtext

`lll` `lllnumbered`

Ein Lücke in einem Text kann mit einem der Befehle

`\lll[Länge der Linie][Ausrichtung]{Lösung}` `\lllnumbered[Länge der Linie][Ausric`

erzeugt werden.

Bei `\lllnumbered` werden die Lösungslinien zusätzlich mit den Buchstaben a, b, c, d, ... y, z gekennzeichnet.

Bei `\lllnumbered` und `\lll` haben die Argumente folgende Bedeutung:

Lösung ist der Text, der auf der Linie dargestellt wird, wenn eines der Argumente `arblsg`, `examlsg`, `kalsg` oder `kamultilsg` (Siehe Seite 9) in der Dokumentklasse angegeben wird.

Länge der Linie ist die Länge der Linie, die anstelle der Lösung dargestellt wird. Die Länge wird als Zahl ohne Einheit angegeben und wird als Millimeter interpretiert.

Wenn keine Länge angegeben wird, dann wird aus der Länge des Wortes `Lösung` eine sinnvolle Länge ermittelt (ca. doppelte Wortlänge).

Es gibt 2 Sonderangaben dieses Arguments

lw setzt die Länge auf die aktuelle Zeilenbreite (**l**ine**w**idth). Damit die Linie nicht über das Zeilenende rausragt, macht dieser Befehl nur am Anfang einer Zeile Sinn.

v lässt die Linienlänge **v**ariabel. Die Linie geht dann vom momentanen Punkt im Dokument bis zum Ende der Zeile. Am Beginn einer Zeile hat das Argument `v` dieselbe Auswirkung wie `lw` benötigt jedoch mehr Rechenleistung.

Innerhalb der `tabbing`-Umgebung kann das Argument `v` nicht verwendet werden.

Ausrichtung gibt an, wie `Lösung` auf der Linie ausgerichtet werden soll. Um den Lösungstext zu sehen muss eines der Argumente `arblsg`, `examlsg`, `kalsg` oder `kamultilsg` in der Dokumentklasse angegeben werden (Siehe Seite 9).

Folgende Werte sind anstelle von `Ausrichtung` zulässig:

c zentriert den Text `Lösung` auf der Linie. Dies ist zugleich das Standardargument.

l setzt den Text der `Lösung` linksbündig auf die Linie.

r setzt den Text der `Lösung` rechtsbündig auf die Linie.

s dehnt (to **s**tretch) den Text `Lösung` so, dass er links- *und* rechtsbündig auf der Linie ist. Dieses Argument hat bei Linien variabler Länge dieselbe Auswirkung wie das Argument c (zentriert).

Zu beachten: `\lll` und `\lllnumbered` haben *zwei* optionale Argumente. Wenn nur ein optionales Argument angegeben wird, dann wird dieses als `Länge interpretiert`. Wenn eine Ausrichtung angegeben werden soll, dann müssen *beide* optionalen Argumente angegeben werden. Soll dabei die Länge der Lösungslinie über die Länge von `Lösung` berechnet werden, ist das Argument leer anzugeben. (z.B. `\lll[][l]{Lösung}`)

Ausgabe der Lösungen als Hilfestellung

Wenn ein Lückentext viele Lücken enthält, dann kann es sinnvoll sein, den Schülern alle als `Lösung` angegebenen Wörter in ungeordneter Reihenfolge als Hilfe zur Verfügung zu stellen.

`lllliste` Dies kann man für alle Lücken die mit `\lllnumbered` erzeugt wurden mit folgendem Befehl tun:

```
\lllliste[Sortierung]
```

`Sortierung` gibt an, in welcher Reihenfolge die Lösungen angegeben werden. Wird für die Sortierung eine `0` angegeben, dann werden die Lösungen in der Reihenfolge ihres Auftretens gelistet.

Bei `1` (Standardargument) und `2` werden die Lösungen durcheinander ausgegeben.

Beispiel für einen Lückentext:

```
\lue Es werden \unit{\lll[20]{}}{\gram} Maschinenöl
von einem Tauchsieder (\unit{300}{\watt}) in \lll[20]{}
Sekunden von \unit{\lll[20]{}}{\celsius} auf
\unit{\lll[20]{}}{\celsius} erwärmt.
```

10.3.3 Mehrzeilige Lücken mit Lösungen

Mehrzeilige Lücken erhält man durch mehrfachen Aufruf von `\lll`.

Beispiel:

```
Ein Auto ist ein Ding, das  \lue \lll[v][l]{fährt}

\lll[lw][l]{wenn man Benzin hat.}

\ll[lw][l]{}
```

Dies erzeugt den Satz `Ein Auto ist ein Ding, das` mit einer anschließenden Lösungslinie bis zum Zeilenende. Danach werden nochmals 2 Zeilen erzeugt, die

über die ganze Seitenbreite reichen.

Beachten Sie, dass mit `\lue` der Absatz auf einen größeren Zeilenabstand Umgestellt werden muss.

Wollen sie In jeder Zeile noch eine Beschriftung voranstellen, und sollen die leeren Zeilen dieselbe Länge haben, so empfiehlt sich die `tabbing`-Umgebung (Tabulator) und die Verwendung von Linien konstanter Länge:

```
\lue
\begin{tabbing}
   Beobachtung:\hspace{8mm} \= \lll[138]{} \\
                            \> \lll[138]{} \\
                            \> \lll[138]{} \\
                                           \\
   Latente Wärme:           \> \lll[138]{} \\
   Sensible Wärme:          \> \lll[138]{} \\
 \end{tabbing}
```

Mit `\lue` wird der Zeilenabstand erhöht. Dann beginnt die `tabbing`-Umgebung.

In der Zeile

```
   Beobachtung:\hspace{8mm} \= \lll[138] \\
```

Wird ein Tabstop (`\=`) 8 Millimeter nach dem Ende des Wortes `Beobachtung` gesetzt. Es folgt eine Linie mit 138 Millimeter Länge und der Sprung in die nächste Zeile (`\\`).

In den nachfolgenden Zeilen wird mit `\>` zum Tabstopp gesprungen, wieder eine Lösungslinie mit 138 Millimeter eingefügt und in die nächste Zeile gesprungen.

10.3.4 Mehrzeilige Lücken ohne Lösungen

Sollen keine Lösungen angegeben werden, sondern nur ein Bereich mit leeren Linien
`lln` nutzt man den Befehl

```
\lln[n]{Länge in mm}
```

Er erzeugt mehrere (`n`) Lösungslinien der Länge `Länge`. Standardlänge ist 161 mm. Es wird eine neue Zeile begonnen.

??? textwidth?, alle Linien Einrücken?

10.4 Lücken außerhalb von Lösungslinien

xlsg

Sollen Lösungen nicht auf eine Linie geschrieben werden sondern in eine Tabelle, unter ein Bild, oder an sonstige Stellen im Text, dann benutzt man den Befehl

```
\xlsg{Lösungstext}
```

Wird in der Dokumentklasse eines der Argumente `arblsg`, `examlsg`, `kalsg` oder `kamultilsg` (Siehe Seite 9) angegeben, dann wird der `Lösungstext` in grüner Farbe sichtbar. Ansonsten ist er unsichtbar, d.h. er wird in weisser Farbe ausgegeben. Der unsichtbare Text nimmt dabei exakt denselben Raum ein wie der sichtbare.

10.5 Erstellen von Stoffverteilungsplänen

Für Stoffverteilungspläne wird das Argument `stoff` in der Dokumentklasse `teacher` benutzt:

```
\documentclass[stoff]{teacher}
\Klasse{M1KB}
\Revision{}
\Ausdruck{}
\begin{document}
\titel
Hier steht einführender Text
\begin{stoff}
  \thema{Wärmelehre}
    \stunde{T}{22.11.04}{Einführung}{Temperatur, ...}{}
    \stunde{M}{27.11.04}{Fortsetzung}{Einheiten und Anwendung}{}
\klassenarbeit
\block
  \thema{Wärmeübertragung}
    \stunde{T}{29.11.04}{Einführung}{Wärmeleitung}{}
    \stunde{M}{05.12.04}{Grunglagen}{Strahlung}{}
\end{stoff}
\end{document}
```

titel

Mit dem Befehl `\titel` kann ein Titel erzeugt werden, der den `namen` der Klasse enthält, der mit `\Klasse{name}` gesetzt wurde.

stoff

Der eigentliche, tabellenförmige Stoffverteilungsplan beginnt und endet mit der Umgebung `stoff`.

Innerhalb der Umgebung `stoff` sind folgende Befehle definiert:

thema

Um eine Zwischenüberschrift in der Tabelle einzufügen nutzen sie den Befehl

```
\thema{Das Thema der nächsten Stunden}
```

`stunde` Eine einzelne Stunde wird mit

```
\stunde{Fach}{Datum}{Zeile 1}{Zeile 2}{Lehrplan}
```

in den Stoffverteilungsplan eingetragen. Für die Fächer M, T, AP werden die Stunden nach Fächern getrennt aufsummiert.

`block` Um das Ende eines Unterrichtsblocks zu markieren, wird der Befehl `\block` eingetragen. Dabei werden die Stundenanzahlen in den Fächern T, M und AP ausgegeben.

`klassenarbeit` Für eine Klassenarbeit können mit diesem Befehl gleich 4 Stunden auf einmal angegeben werden, die zur Wiederholung, Durchführung und Besprechung genutzt werden.

11 Vorgefertigte Texte für Kältetechnik

Befehl	Etwaiges Aussehen
`\hxdia`	h,x-Diagramm
`\hlogpdia`	h,log p-Diagramm
`\pvdia`	p,V-Diagramm

`hxdia` `hlogpdia` `pvdia`

12 Einbinden von Tabellenkalkulationsdaten

`problectix` kann Daten aus Zellen eines Tabellenkalkulationsblattes einlesen. Dabei werden die errechneten Werte einer Zelle (nicht die Formel) eingebunden.

Die Tabellkalkulationsdatei muss im Excel-97 Format vorliegen. Das Auslesen von OpenOffice Calc Dateien (*.ods) wird noch nicht unterstützt, da es für ein solches Perl-Modul unter Ubuntu noch kein fertiges Paket gibt.

Um nicht auf Microsoft-Produkte zurückgreifen zu müssen ist folgender Workflow sinnvoll:

1. Erstellen der Tabellenkalukulation mit OpenOffice.
2. Speichern unter *.xls (Excel-97 Format)
3. Importieren der Daten in `problectix` mit untenstehenden Befehlen.

Nach einer Änderung im *.ods-Format, muss dann erneut im Excel-Format abgespeichert werden.

12.1 Befehle zum Einbinden von *.xls Tabellen

Alle Befehle die auf **sp**read**sh**eet-Dateien zugreifen beginnen mit `\spsh`.

Zuerst gibt man die einzubindenede *.xls-Datei an (relative Pfadangabe):

```
\spshfile{Projekt-Kuehlzelle.xls}
```

Dann kann man einstellen, wieviele Nachkomma-Stellen angezeigt werden sollen (Die `spsh`-Befehle lesen immer die volle Genauigkeit aus der Zelle und ignorieren Formatierungen in der Zelle). Standardwert ist 2. Von dem Befehl sind nur Zahlen betroffen.

```
\spshdigits{3}
```

Manchmal ist es notwendig, eine Zahl als String zu interpretieren. Dann kann man mit

```
\spshstring
```

vor dem Zellenbefehl (siehe unten) umschalten. Und mit

```
\spshnostring
```

wieder zurückschalten. Die eingestellte Nachkommazahl bleibt erhalten.

Dann kann man auf die Zelle C6 in der Tabelle 1 in dieser Datei zugreifen mit:

```
\spshcell{1}{C6}
```

will man gleich mehrere Zellen (C6 bis C12) auslesen, um mehrere Gruppen zu erstellen, erfolgt dies mit

```
\spshcells{1}{C6:C8}
```

Dieser Befehl ist identisch mit:

```
\abc{\spshcell{1}{C6}}%
    {\spshcell{1}{C7}}%
    {\spshcell{1}{C8}}
```

Sobald ein Befehl `\spshcells`-Befehl auftritt, werden beim Übersetzen mit jefflatex die Dokumente aller Gruppen erzeugt, und in der PostScript-Datei hintereinandergehängt.

Teil III

Allgemeine Tipps

13 Grundlegende Dinge

13.1 Einheiten

Einheiten werden mit dem Paket `SIunits` erstellt. Dieses Paket wird von der Dokumentklasse `teacher` automatisch geladen.

13.1.1 Einheitenschreibweise

Einheiten-Erläuterungen werden so dargestellt:

`$[F] = \newton$` ergibt: $[F] = \mathrm{N}$

13.1.2 Besondere Einheiten

Ein Gradzeichen wird erzeugt mit:

`\celsius`	ergibt °C
`\textdegree`	ergibt °
`\degree`	ergibt °

13.2 Hilfsbefehle

`frage` Mit `\frage{Fragestellung}` wird `Fragestellung` sehr auffallend im umgebenden Text dargestellt.

13.3 Schriftart/Sonderzeichen

`entspricht` Der Befehl `\entspricht` erzeugt ein mathematisches *entspricht*-Symbol

13.3.1 Aus LaTeX und anderen Paketen

`textmu` Der Befehl `\textmu` erzeugt ein Mikrometer-Zeichen (nichtkursives µ). Bei Zahlenangaben sollte jedoch `\unit{200}{\micrometre}` verwendet werden.

`textbackslash` Der Befehl `\textbackslash` erzeugt einen Backslash

`textperthousand` Der Befehl `\textperthousand` erzeugt ein Promille-Zeichen passend zum Prozentzeichen.

`textvisiblespace` Unsichtbare Leerzeichen (hinter einem Befehl) werden erzeugt mit `\{}`. Sichtbare Leerzeichen mit `\textvisiblespace`

`varnothing` Der Befehl `$\varnothing$` erzeugt einen Durchmesser/Durchschnitt-Zeichen. Dazu ist das Package `amssymb` notwendig, das von der Dokumentklasse `teacher` automatisch geladen wird.

Schöner als Klassendurchnittszeichen ist das durchgestrichene O:

`\O{}`	ergibt	Ø
`\o{}`	ergibt	ø

`\EUR` `\EURtm` `\EURhv` `\EURcr` Die Makros `EUR`, `EURtm`,`EURhv` und `EURcr` erzeugen Euro-Symbole für normale?? Schrift, Times-Schriften, Helvetica-Schriften bzw. Courier Schriften.

Notwendig dazu ist das Package `marvosym`

13.4 Für den Lehrer

??? Flexibler: Text, Länge variabel, links/rechts/mittig

`unterschrift` Mit dem Befehl `\unterschriftift[Text]` wird ein Leerraum für eine Unterschrift erzeugt. Unter dem Unterstrich steht als standardmäßiger Text: `(Beck) Klassenlehrer im BVJA`. Dies kann mit dem optionalen Argument `Text` verändert werden.

13.5 Tasten der PC-Tastatur

Vorraussetzung ist, dass `\usepackage{fancybox}` im Vorspann steht und die Datei `fancybox.sty` auf dem PC vorhanden ist.

Befehle müsssen erst noch geschrieben werden.

14 Befehle für das Fach Deutsch

`\wort` Der Befehl

```
\wort[Abstand]{Zu erklärendes Wort}{Beispiel}{Erklärung/Übersetzung}
```

erzeugt Worterklärungen in unterschidlicher Formatierung. Geeignet um Begriffe oder Fremdwörter zu erläutern. Nach jeder Erklärung sollte in eine neue Zeile gesprungen werden, da ein vergrößerter Zeilenabstand eingefügt wird.

Mit dem Zusatzargument `Abstand` wird der Abstand vor und nach der Worterklärung beeinflusst. Wird bei jedem Befehl derselbe Abstand angegeben, so erfolgt ein normaler Zeilenumruch mit dem zuätzlichen Abstand `Abstand`. Ist der Abstand 0mm, fügt sich die Erklärung problemlos in Fließtext ein.

`konjugation` Der Befehl `\konjugation{Verb}{in der Zeitform}` erzeugt eine auszufüllende Konjugationstabelle.

Die Argumente bedeuten:

`Verb` ist das zu konjugierende Verb. Es wird unterstrichen dargestellt.

`in der Zeitform` ist die Fortsetzung des Satzes: Konjugieren Sie das `Verb` ...

`bspsatz` Der Befehl `\bspsatz[Bilden Sie Sätze!]{Bspsatz}{Lösung}` erzeugt einen umrahmten Kasten, mit Aufgabenstellung, Aufgabe und Lösung.

Die Argumente bedeuten:

`Bilden Sie Sätze!` ist die Standardmäßige Aufgabenstellung. Sie kann im optionalen Argument verändert werden.

`Bspsatz` gibt die Beispiel-Aufgabe vor.

`Lösung` gibt die Musterlösung vor. Sie erscheint auf der Lösunglinie.

`bspzweisatz` Der Befehl

```
\bspzweisatz[Bilden Sie Sätze!]%
            {Bspsatz}{Info 1:}{Lsg 1}{Info 2:}{Lsg 2}
```

erzeugt einen umramten Kasten, mit Aufgabenstellung, 2 Aufgaben und 2 Lösungen.

Die Argumente bedeuten:

`Bilden Sie Sätze!` ist die Standardmäßige Aufgabenstellung. Sie kann im optionalen Argument verändert werden.

`Info 1,2` machen weitere Angabenvor der Musterlösung.

`Lsg 1,2` geben die Beispiel-Lösungen vor. Sie erscheinen auf den Lösunglinien

15 Zählerdateien

Der Inhalt der Zähler-Dateien soll am Beispiel von `ka-format.tex` erläutert werden. Diese Datei wird eingelesen, wenn das Argument `[ka]` in der Dokumentklasse `teacher` angegeben wird.

Die Datei `ka-format.tex` beinhaltet folgende Zähler:

`loesungslinienzeigen` Zeigt Lösungslinien (1) oder verbirgt sie (0)

`teilaufnummerierung` Art der Nummerierung der Aufgaben: a,b,c... (=0) oder 1.1,1.2,1.3...(=1)

`aufgabenpunkte` Punktzahl zeigen = (1), verbergen = (0). Gemeint ist die Punktzahl am Ende einer ganzen Aufgabe.

`aufgabenkopfzeile` Schaltet die Ausgabe der Aufgaben-Kopfzeile ein = (1), aus = (0).

`aufgabennummerierung` Schaltet die Nummerierung der Aufgaben auf T :(1), oder auf Aufgabe :(0).

`aufgabenstellung` Aufgabenstellung (Fragen) zeigen = (1), verbergen = (0).

`aufgabenfusszeile` Schaltet die Ausgabe der Aufgaben-Fusszeile ein = (1), aus = (0).

`loesungkopfzeile` Schaltet die Ausgabe der Lösungs-Kopfzeile ein = (1), aus = (0).

`loesungen` Lösungen (Antworten) zu den Aufgaben zeigen = (1), verbergen = (0).

`xlsg` Farbe des Lösungstextes auf grün = (1), verbergen (Weiß) = (0).

`dateinamen` Dateinamen in der Aufgaben-Kopfzeile zeigen = (1), verbergen = (0).

`dehnen` Variable Zwischenabstände ja = (1), nein = (0) (nicht vollständig implementiert)

`fachangabe` Fach der Aufgabe im Aufgabentitel-Kopf zeigen = (1), oder aus = (0)

`fachangabepzk` Fachangabe-Buchstaben (M, T, AP) am Punktzahl-Kasten zeigen = (1), oder aus = (0)

`punkteangabepzk` Erreichbare Punktezahl der Teilaufgabe am Punktzahl-Kasten zeigen = (1), oder aus = (0)

`punktzahlkasten` Punktzahl-Kasten zeigen = (1), oder aus = (0)

`punktesummezeigen` Punktesumme der Teilaufgaben am Ende der Aufgaben zeigen = (1), oder aus = (0). Veraltet.

`aufgabentitel`	Name der Aufgabe hinter der Aufgabenmarke im Aufgabentitel-Kopf zeigen = (1), oder aus = (0)
`gruppeninfo`	Informationen über die Gruppen in der Aufgaben-Fusszeile zeigen. aus = (0), SW = (1) oder in Farbe = (1) ????? Farbe/SW wird mit bw
`gruppeninfohead`	Informationen über die Gruppen im Aufgaben-Kopf zeigen. aus = (0), SW = (1)
`notenliste`	Erzeugen einer Noteliste am Ende des Dokuments. aus = (0), zeigen = (1)
`lkaabstand`	Längenangabe. Zusatzabstand der Lösungslinien für den lka-Befehl. kommt zum Zeilenumbruch noch hinzu???
`zusatzlkaabstand`	Längenangabe. Zusätzlich zur Länge lkaabstand vor der ersten Lösungslinie eingefügt, um Aufgabe und Lösung klarer zu trennen.

Teil IV

DIN A 5 Blätter

Wenn man nur ein kleineres Arbeitsblatt oder einen Test erstellen will ist oft DIN A 5 ausreichend. Da aber alle Drucker bzw. Kopierer üblicherweise mit DIN A 4 Blättern arbeiten hat man einige Probleme, die einen sehr schnell wieder Abstand nehmen lassen vom erstellen von DIN A5 Blättern.

`problectix` löst diese Probleme mit

Teil V

Lernkarten (Flashcards)

16 Einführung

Um Wissen auswendig zu lernen, sind Lernkarten (Vorderseite: Frage, Rückseite: Antwort) sehr gut geeignet. In ihrer modernen, digitalen Form sind sie auf portablen Geräten (Handys, Tablets) allerorts einsetztbar.

17 Anki

Die Software `Anki` gibt es für Linux, FreeBSD, Windows, Mac OSX, Android (AnkiDroid) und iOS.

Man kann mit `Anki` sowohl Lernkarten erstellen als auch Lernkarten lernen.

Beim Erzeugen der Lernkarten ist es möglich, LaTeX-Code in die Lernkarten einzubetten, der dann in verlinkte Grafiken umgewandelt wird (LaTeX-Installation und Konfigurationsanpassungen von Anki erforderlich). Damit wird es möglich Formeln, Phonetische Zeichen, ... darzustellen.

Beim Lernen der Karten werde Informationen zum Lernfortschritt gespeichert (Wann zu wiederholen, gewusst oder nicht, ...), sodass gezielt weiter gelernt werden kann.

Problematisch dabei ist, dass die beim erstellen von Lernkarten mit `Anki` entstehende `*.anki`-Datei eine SQLite-Datenbank ist, die sowohl den Inhalt der Lern-

karten enthält (Frage, Antwort, Tags) als auch Informationen über den jeweiligen Lernstand der Karte. Würde man eine solche SQLite-Datei unter Versionsverwaltung stellen, wäre es nur mit großer Mühe möglich mit Hilfe von `diff` die Entwicklung des Inhalts der Karten sowie Fehlerkorrekturen nachzuvollziehen.

Geeigneter für Versionsverwaltung wäre das Textformat, dass `Anki` für den Import bzw. Export-Format nutzt (Eine Zeile je Lernkarte, Semikolonsepariert Vorne;Hinten;Tags). Leider wäre es dann notwendig für eine LaTeX-Vorschau diese Textdatei in `Anki` zu importieren. Eine schnelle Vorschau wäre also unmöglich.

Außerdem wäre es wünschenswert den LaTeX-Inhalt mit einem ordentlichen Editor zu erstellen (Tastaturkürzel, Syntax-Highlighting, ...).

Diese Probleme werden in `problectix` beseitigt.

18 Anki innerhalb von `problectix`

Es wird als Quelle das Dateiformat `anki-latex-source` eingeführt. Dies ist eine LaTeX–Datei mit der Endung `.tex`. Sie wird unter Versionsverwaltung gestellt und kann mehrere Lernkarten enthalten.

Diese Datei enthält in Kommentaren folgende Steueranweisungen für das Übersetzen mit `jefflatex`:

1. Anweisungen, die nur einmal vorkommen dürfen:

   `%anki` Der `Anki`-Karten-Stapel beginnt

   `%end` Der Karten-Stapel endet

2. Für jede Lernkarte *müssen* dann folgende Anweisungen in der angegebenen Reihenfolge vorkommen:

   `%front` Jetzt folgt die Vorderseite einer (neuen) Karte.

   `%back` Jetzt folgt die Rückseite der Karte.

   `%cat cat1 cat2` Die zuvor beschriebene Karte ist in der Kategorie `cat1` und `cat2`

Beim Aufruf von `jefflatex` (Taste F5 in emacs) werden die Anweisung dann umgesetzt (Vorspann, Nachspann, Neue Seite, ...) und *zwei* Dateien erzeugt:

1. Eine temporäre LaTeX-Datei, deren Kompilat als Vorschau angezeigt wird (1 Lernkarte je Seite, Frage und Antwort durch einen breiten horizontalen Strich getrennt, in Zukunft evtl. nebeneinander).

2. Außerdem wird eine `*-anki.txt`-Datei erzeugt, die den LaTeX-Code so enthält, wie ihn `Anki` beim Import erwartet (3 Felder je Zeile (Frage;Antwort;Kategorien).

So kann das in der Vorschau gezeigte Layout jederzeit in `Anki` importiert werden.

Im Moment ergeben sich (noch) folgende Einschränkungen:

- Vorderseite bzw. Rückseite muss *komplett* in LaTeX erzeugt werden.
- Als Vorspann kann nur der im Paket eingebaute benutzt werden. Und nicht der in `.anki` befindliche und anpassbare, den `Anki` nutzt.
- Anführungszeichen (solche die mit Shift-2 erzeugt werden) machen Probleme, da sie von `Anki` speziell interpretiert werden. Am besten nimmt man Englische Anfürungszeichen:
  - Beginnendes Anführungszeichen: Zweimal Backticks (neben der Delete Taste, Shift gedrückt halten)
  - Endendes Anführungszeichen: Zweimal die Taste, auf der auch # liegt, Shift gedrückt halten
- In `Anki` wird jede Kartenseite in einer *eigenen* LaTeX-Datei kompiliert. Es kann also in `Anki` keine Kartenseitenübergreifenden Variablen, Macros, ... geben, obwohl dies in der Vorschau umsetzbar ist (Eine LaTeX-Datei für den ganzen Stapel).

19 Tipps zu Anki

19.1 Starten von Anki

`Anki` wertet bei Benutzung von LaTeX-Code die Umgebungsvariable `TEXINPUTS` aus, was dazu führt, dass beim Erstellen von Karten evtl. die falschen Grafiken gefunden und angezeigt werden. Deshalb sollte `Anki` mit dem Befehl:

```
problectix-anki
```

gestartet werden, der `TEXINPUTS` vor dem start leert.

In Ubuntu Unity findet sich dazu ein Icon/Starter im Dash (Anki-Problectix). Das Icon ist dasselbe wie bei Anki, hat aber rote Sterne statt blaue.

19.2 Verarbeiten des Quellcodes in Anki 1.2.x

Folgendes Vorgehen ist erforderlich:

1. Starten von `Anki-problectix` (Konsole: `problectix-anki`).
2. Datei $\longrightarrow$ Importieren ...
3. Dem Kartenstapel einen sinnvollen Namen geben.
4. `*-anki.txt`-Datei auswählen.
5. Importieren (Fenster):
   (a) Schraubenschlüssel-Button hinter `Modell:`-Box klicken um Stapeleinstellung für den Import zu wählen.
   (b) Auf den Reiter `LaTeX` gehen.
   (c) Dort *nach der Zeile* `\usepackage{amssymb,amsmath}`
   Die Zeile `\input{/usr/share/problectix-anki/latex/preamble-input.tex}` einfügen.
   Oder besser noch:
   `\InputIfFileExists{/usr/share/problectix-anki/latex/preamble-input.tex}{}{p`
6. Abschließen durch Klick auf den Button `Importieren`.
7. Es sollte die Meldung xy Fakten aus *-anki.tex importiert erscheinen.
8. Fenster Schließen.
9. Lernen.

Verbesserung: Das postinstall script vom Paket `problectix-anki` soll `/usr/share/anki/deck.py` so patchen, dass die `InputIfFileExists-` Zeile mit Kommentar da drin steht.

19.3 Verarbeiten des Quellcodes in Anki 2.0.x

Teil VI

Die Perl-Scripte von problectix

20 Einführung

Diese Scripte helfen die erstellten Aufgaben zu verwalten.

Sie können in der Datei `/etc/problectix/problectix.conf` konfiguriert werden.

20.1 problectix-test

Mit `problectix-test` kann das korrekte Funktionieren von `problectix` getestet werden. Dazu wird im Homeverzeichnis des users ein Verzeichnis `problectix-test` erzeugt, in das Beispieldateien kopiert werden.

Diese Beispieldateien werden mit `latex`, `dvips` bzw. `pdflatex` sowie `pstops` verarbeitet.

Schließlich können sie mit einem Dateibetrachter visuell geprüft werden.

Alle aktuellen Optionen von `problectix-test` können sie mit

```
problectix-test --help
```

anzeigen lassen.

20.2 jefflatex

`jefflatex` übersetzt die im LaTeX-Format erstellen Aufgaben nach PostScript (*.pd), PDF (*.pdf) oder PNG (*.png). Nähere Informationen erhalten sie im Konsolenfenster mit:

```
man jefflatex
```

20.3 problectix

Mit dem Befehl

```
problectix --www --browsetree
```

Können Vorschaubilder der Aufgaben erzeugt werden, die sich im Homeverzeichnis in `problectix-data` befinden. Mit

```
man jefflatex
```

findet man die Dokumentation zu diesem Befehl.

20.4 `einmaleins`

20.5 `problectix-marklist`

20.6 `treadmillix`

Not in the package at the moment.

Teil VII

Erstellen von Vektorgrafiken

21 Technische Zeichnungen mit `librecad`

21.1 Einbinden von technischen Zeichnungen

Zum Einbinden von technischen Zeichenungen ist das Programm `librecad` ab Version 2.0 geeignet.

Gehen Sie folgendermaßen vor:

1. Erstellen sie die LibreCAD-Zeichnung. Speichern Sie die Datei im dxf-Format ab.
2. Gehen Sie in LibreCAD auf die Druckvorschau. Wählen sie evtl. den Knopf für Schwarz-Weiss-Darstellung aus, falls sie die Zeichnung in Schwarz-Weiss einbinden wollen.
3. Drucken Sie dann die Vorschau in eine Datei.

   Da hier PostScript erstellt wird, sollten sie dieser Datei die Endung `*.ps` geben. Diese Datei ist ein *ganzseitiges* PostScript-Dokument. Sie können es sich zum Beispiel mit

   ```
   kghostview datei.ps
   ```

   anschauen.
4. Um nur die Zeichnung (ohne weissen Rand) in eine Datei zu einzufügen, muss diese PostScript-Datei in *Encapsulated PostScript* umgewandelt werden. *Encapsulated PostScript* besitzt eine sogenannte BoundingBox, die angibt welcher Ausschnitt der ganzseitigen PostScript-Datei genutzt werden soll.

   Die Umwandlung geschieht mit dem Befehl (Man Lese: ps-to-epsi):

   ```
   ps2epsi datei.ps
   ```

   Dabei entsteht eine Datei `datei.epsi` die das Format von *Encapsulated PostScript* hat. Diese Datei können sie sich zum Beispiel mit `kghostview Dateiname` anschauen. Beachten Sie, dass der Rand automatisch beschnitten wurde.

   Falls sie mit der Beschneidung nicht einverstanden sind, können sie in der Datei `datei.epsi` die BoundingBox mit einem Texteditor anpassen.

5. Fügen sie die *Encapsulated PostScript*-Datei in ihr LaTeX-Dokument ein mit dem Befehl:

   ```
   \includegraphics{datei.epsi}
   ```

21.2 Maßstabgetreues Einbinden

In vielen Fällen soll eine in einem bestimmten Maßstab erzeugte technische Zeichnung in exakt derselben Größe oder in einem bestimmten Maßstab eingebunden werden.

Dann können die Schüler z. B. Maße aus der Zeichnung herausmessen, mit Zeichenschablonen Symbole ergänzen usw.

Beim maßstabgetreuen Einbinden geht man so vor:

1. In der Druckvorschau von `librecad` wählt man als Maßstab 1:1 aus. Mit dieser Einstellung erzeugt man eine PostScript-Seite, in der alle Längen exakt so sind, wie mit `librecad` gezeichnet.
2. Mit `ps2epsi datei.ps` wird die PostScript-Seite auf eine unbestimmte Größe beschnitten. Die Originalgröße ist jedoch immer noch in der Datei vermerkt.
3. Zum Einbinden in ihr LaTeX-Dokument wählen sie folgenden Befehl:

   ```
   \includegraphics[scale=1]{datei.epsi}
   ```

   Es können natürlich auch andere Maßstäbe angegeben werden, um das Bild um einen bestimmten Faktor zu vergrößern oder zu verkleinern.

21.3 Sonstiges

Option bw bzw. sw nutzen, wenn Schwarzweiss und Farbige Vektorgrafiken existieren???

22 Vektorgrafiken mit `xfig`

Mit `xfig` kann man Vektorgrafiken erstellen, deren Textfelder mit beliebiegen LaTeX-Befehlen ersetzt werden können.

So ist es möglich in einer Grafik Anstelle einer komplexen Formel den Text `waermeleistung` einzufügen, und diesen dann mit LaTeX-Code zu ersetzen, indem man *vor* dem Einbinden der Grafik (Siehe Seite 21) einfügt:

```
\psfrag{waermeleistung}{$\dot{Q}$}
```

Dies ersetzt in der Grafik den Text `waermeleistung` mit $\dot{Q}$. Mit dem obigen `\psfrag`-Befehl werden *alle* Stellen, an denen `waermeleistung` steht, ersetzt.

Mehr Informationen zum Paket `psfrag` in dessen Dokumentation.

23 Vektorgrafiken mit `dia`

Beim exportieren darauf achten, dass als *Gekapseltes PostScript (Pango-Schriften* exportiert wird.

24 Vektorgrafiken mit `scribus`

Jedes Programm mit Druckfunktion kann eine PostScript-Datei erzeugen, indem man einen PostScript-Drucker einrichtet und dann in eine Datei druckt.

Mit `scribus` kann man diese `*.eps`- bzw. `*ps`-Dateien einlesen und weiterverarbeiten (Beschriften, Teile Löschen, ...).

Dazu geht man in `scribus` wie folgt vor:

1. `Datei` $\longrightarrow$ `Importieren` $\longrightarrow$ `EPS/PS-Dateien importieren ...`
2. Bearbeiten, ...
3. `Datei` $\longrightarrow$ `Exportieren` $\longrightarrow$ `Seite als EPS speichern ...`

Beispiel für Anwendungen:

- Screenshots beschriften.
- Ausdrucke von `coolpack` kommentieren, Lösungen einzeichnen.
- ...

25 Weitere Informationen

Dokumentation kann man sich mit folgenden Befehlen anzeigen lassen.

Diese vorliegende Dokumentation

```
texdoc problectix
```

Dokumentation zum Paket `SIunits`

```
texdoc problectix
```

Dokumentation zum Paket `psfrag`

Tut nicht

```
texdoc psfrag
```

Teil VIII

Einrichten einer Arbeitsumgebung

Im Gegensatz zur anderen Textverarbeitungsprogrammen wie Openoffice Writer oder Microsoft Word, besteht ein LaTeX-Textsatzsystem aus mehreren verschiedenen Komponenten, die zum Zusammenarbeit gebracht werden müssen.

Im folgenden wird zuerst `eclipse` besprochen, das diese Komponenten zu einer sogenannten Entwicklungsumgebung integriert (IDE: integrated development environment)

Anschließend werden Programme besprochen, die eine Teilfunktion übernehmen.

26 Arbeiten ohne X

Installieren sie den ps-Viewer `bmv`. lassen sie sich PostScript-Dateien anzeigen mit:

```
bmv -r400x400 datei.ps
```

27 Aufgabenübersicht erstellen und ansehen

27.1 Aufgabenübersicht erstellen

Bei einer großen Aufgabensammlung ist es umständlich, sich Aufgaben in einzeln anzeigen zu lassen.

Eine Übersicht mit Vorschaubildern *aller* Aufgaben im Verzeichnis `mytex???` kann erstellt werden mit dem Befehl:

```
problectix --browsetree
```

Dies erzeugt eine `html`-Seite mit Vorschaubildern (`*.png`) jeder Aufgabe aus dem Verzeichnis `mytex???` im Vorschauverzeichnis `problectix-tree`.

Je nach Anzahl der Aufgaben kann der Befehl sehr lange dauern.

27.2 Aufgabenübersicht ansehen

Die Vorschaubilder können sie nun mit einem Browser (z.B. `firefox`) anschauen(Siehe Abbildung 1).

Geben sie dazu folgende URL ein:

```
file:///home/user/problectix-tree/html-lehrer/index.html
```

Der Teil `/home/user` gibt das Verzeichnis ihrer privaten Daten an und kann variieren. Am besten Speichern sie diese URL in ihren Bookmarks ab.

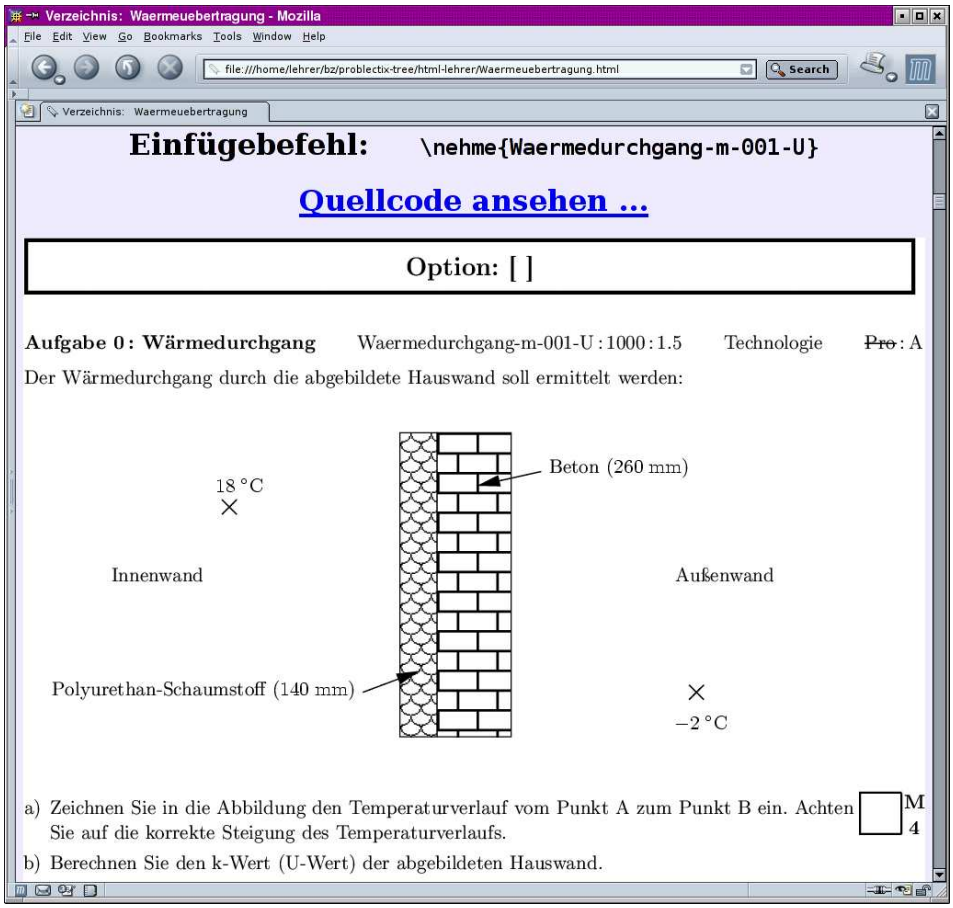

Abbildung 1: Bildvorschau mit `mozilla (firefox)`

Durch Klick auf `Quellcode ansehen ...` können sie den Inhalt der `*.tex`-Datei ansehen.

27.3 Klassenarbeit erstellen

Zum erstellen einer Klassenarbeit öffnen sie 2 Programme:

1. *firefox* zum ansehen uind auswählen der Aufgaben (siehe 27.2).
2. Einen Editor wie z.B. `emacs`.

Wenn sie in `firefox` eine Datei sehen, die Sie in der Klassenarbeit benutzen wollen, dann kopieren sie den `Einfügebefehl` in den Editor[1]. Der Einfügebefehl beginnt mit `\nehme`.

Um aus den Aufgaben eine Klassenarbeit zu machen, müssen sie von Hand im Editor noch folgende Zeilen ergänzen:

```
\documentclass[ka]{teacher}

\begin{document}

\nehme{Aufgabe-1}

\nehme{Aufgabe-2}

\end{document}
```

Zum Ausfüllen des Kopfes siehe Seite 14.

Für weitere Dokumnttyp-Argumente siehe Seite 9.

[1] Unter Linux (genauer gesagt unter `X Window`) ist das Kopieren sehr einfach: Markieren sie mit der linken Maustaste. Der markierte Text ist nun schon in der Zwischenablage. Fügen sie diesen Text mit der mittleren Maustaste ein.