»Flotter Oldie, LaTeX – 25 Jahre und kein Ende«, lautete die Überschrift eines Beitrags in der Computerzeitschrift c't, was jetzt schon wieder mehr als sechs Jahre her ist. [48] Die TeX Users Group (`http://tug.org`) setzte ihre Jahrestagung 2010 unter das Motto »TeX's 2^5 anniversary« (`http://tug.org/tug2010/`). Man hat es also mit einer ziemlich alten Software zu tun, die man auch getrost als Dinosaurier der Programmentwicklung bezeichnen kann. Anfänglich nur für Mathematiker gedacht, kann heutzutage faktisch jeder das Programm TeX benutzen und qualitativ hochwertige Dokumente erstellen. Dabei wird es im Allgemeinen eine Anwendung von LaTeX sein, welches von Leslie Lamport entwickelt wurde, um das von Donald Knuth geschaffene TeX anwenderfreundlich zu gestalten.

Während die Mathematiker eigentlich weltweit nur LaTeX als eine adäquate Möglichkeit betrachten, ihre mit Formeln gespickten Texte anspruchsvoll zu setzen, sind im letzten Jahrzehnt fast alle anderen Fachbereiche dazugekommen. Sie alle vereint die Überzeugung, dass Dokumente, egal von welchem Umfang und welcher Komplexität, mit LaTeX garantiert mehr als nur zufriedenstellend gesetzt werden können.

Die Veränderungen im Druckwesen haben den Beruf des Setzers überflüssig werden lassen. Dafür entstand die digitale Druckvorstufe, weshalb nun ein jeder Autor, Lektor und Verlag in einer Person werden kann. Und Kleinauflagen sind mittlerweile finanziell auch nicht mehr das große Problem. Mit LaTeX lassen sich qualitativ hochwertige PDF-Dokumente erstellen, welche faktisch von der digitalen Druckvorstufe »durchgewunken« werden können und dann als gedrucktes Buch oder Zeitschrift erscheinen.

1.1 TeX

TeX kann für sich in Anspruch nehmen, das letzte Softwarepaket zu sein, welches sowohl aus den Anfängen der Unix- als auch der Linux-Ära übrig geblieben ist. Und nicht nur das, es ist mit seinen nun mehr als 32 Jahren aktuell wie nie zuvor. Dies liegt auch an der Komplexität des Textsatzes an sich und der Umsetzung in ein Programm. Es wurden mehrere Versuche unternommen, einen würdigen Nachfolger für TeX zu entwickeln, der die heutigen Möglichkeiten an Soft- und Hardware besser nutzen kann. Um manche Dinge im Programmcode von TeX zu verstehen, muss man sich vergegenwärtigen, dass ein damals zur Verfügung stehender Speicher von circa 100 kByte eben algorithmische Tricks implizierte, die heute nicht mehr notwendig wären. Es bleibt die Hoffnung, dass die letzte aktuelle, auf TeX aufbauende Neuentwicklung, nicht ebenso scheitern wird, wie alle anderen. Mit LuaLaTeX kann man immerhin schon alltagstaugliche Dokumente erstellen.

1.2 Textverarbeitung versus Satzprogramm

TeX ist keine Textverarbeitung im eigentlichen Sinne. Dies wird schon daran deutlich, dass es völlig unerheblich ist, mit welchem System der Quelltext für das *Programm* TeX erstellt wird. Der Anwender programmiert, d. h. er erstellt ein Programm, welches als Ausgabe einen formatierten Text ausgibt. Es erleichtert Anfängern den Umgang mit dem

Gesamtsystem, wenn man sich auch als *Programmierer* versteht. Dies bedeutet, dass es völlig unwichtig ist, welche Formatierung der Quelltext aufweist, solange man die Regeln der dem System zugrunde liegenden Befehlssyntax beachtet. Dies betrifft primär den Zeilenumbruch; ein Satzprogramm umbricht grundsätzlich Absätze, wohingegen fast alle Textverarbeitungsprogramme Zeilen umbrechen. Abbildung 1.1 zeigt den Umbruch einer Textverarbeitung; diese beziehen den Umbruch nur auf eine Zeile, indem diese immer dann beendet wird, wenn nicht mehr genug Platz für ein folgendes Wort vorhanden ist. Dies führt dazu, dass die zweite Zeile mit einem sehr großen Wortabstand gesetzt wird, was optisch immer sehr zweifelhaft erscheint.

Neithardt von Gneisenau,

der Kommandant der Festung Kolberg, deren ruhmreiche Verteidigung einen der interessantesten Abschnitte dieses Buches (von http://www.gutenberg.org/files/23333/23333-8.txt) bildet, 1760 geboren, hatte schon in einem zu Erfurt garnisonierenden österreichischen und danach in einem der Regimenter des Markgrafen von Ansbach-Bayreuth gedient, die in englischem Solde in und gegen Amerika kämpften, als Friedrich der Große ihn 1786 als Premierleutnant in die preußische Armee aufnahm.

Abbildung 1.1: Zeilenumbruch einer Textverarbeitung (OpenOffice)

Ungünstige Zeilenumbrüche

Bei einer Textverarbeitung erfolgt *nach* einem Zeilenumbruch unabhängig von den zukünftigen Zeilenumbrüchen keine Änderung mehr. Diese Tatsache führt immer dann zu ungünstigen Umbrüchen, wenn die Zeile Bestandteile enthält, die nicht getrennt werden sollen oder können. In dem angegebenen Beispiel ist es eine lange Internetadresse, die ohne externen Eingriff nicht getrennt wird. Vergleicht man dazu Abbildung 1.2, welche die Ausgabe von TEX darstellt, so fällt sofort auf, dass sich die Größe der Lücken verringert hat.

Neithardt von Gneisenau,

der Kommandant der Festung Kolberg, deren ruhmreiche Verteidigung einen der interessantesten Abschnitte dieses Buches (von http://www.gutenberg.org/files/23333/23333-8.txt) bildet, 1760 geboren, hatte schon in einem zu Erfurt garnisonierenden österreichischen und danach in einem der Regimenter des Markgrafen von Ansbach-Bayreuth gedient, die in englischem Solde in und gegen Amerika kämpften, als Friedrich der Große ihn 1786 als Premierleutnant in die preußische Armee aufnahm.

Abbildung 1.2: Absatzumbruch eines Satzprogramms (TEX).

Absatzumbruch

Da bei TEX *immer* der gesamte Absatz für den Umbruch der einzelnen Zeilen herangezogen wird, kann das Aufeinandertreffen von eng und sehr weit gesetzten Zeilen vermieden werden; jeder Umbruch steht in Bezug zum vorhergehenden und nachfolgenden Umbruch. TEX kennt vier optische Kriterien für den Satz von Zeilen: *eng*, *weniger eng*, *weit* und *sehr weit*. Nach Definition dürfen aufeinander folgende Zeilen nur benachbarte Kriterien erfüllen, beziehungsweise die gleiche Anordnung aufweisen. Dadurch ist es bei TEX unmöglich, dass eine sehr weit gesetzte Zeile einer eng gesetzten folgt, was

sonst zu den unangenehmen Lücken in Abbildung 1.1 auf der vorherigen Seite führen würde.

Das optische Erscheinungsbild kann weiter verbessert werden: Zum einen durch eine automatische Umbruchmöglichkeit der Internetadresse (Abbildung 1.3) und zum anderen durch einen zusätzlichen Einsatz des Paketes `microtype` (Abbildung 1.4), welches Verfahren der Mikrotypografie anwendet (siehe Abschnitt 5.9.3 auf Seite 171).

Neithardt von Gneisenau,

der Kommandant der Festung Kolberg, deren ruhmreiche Verteidigung einen der interessantesten Abschnitte dieses Buches (von `http://www.gutenberg.org/files/23333/23333-8.txt`) bildet, 1760 geboren, hatte schon in einem zu Erfurt garnisonierenden österreichischen und danach in einem der Regimenter des Markgrafen von Ansbach-Bayreuth gedient, die in englischem Solde in und gegen Amerika kämpften, als Friedrich der Große ihn 1786 als Premierleutnant in die preußische Armee aufnahm.

Abbildung 1.3: Absatzumbruch mit Linkumbruch (LaTeX).

Neithardt von Gneisenau,

der Kommandant der Festung Kolberg, deren ruhmreiche Verteidigung einen der interessantesten Abschnitte dieses Buches (von `http://www.gutenberg.org/files/23333/23333-8.txt`) bildet, 1760 geboren, hatte schon in einem zu Erfurt garnisonierenden österreichischen und danach in einem der Regimenter des Markgrafen von Ansbach-Bayreuth gedient, die in englischem Solde in und gegen Amerika kämpften, als Friedrich der Große ihn 1786 als Premierleutnant in die preußische Armee aufnahm.

Abbildung 1.4: Absatzumbruch mit Linkumbruch und Anwendung der Mikrotypografie (LaTeX).

Der Umbruch für den Absatz wird nach dem so genannten Best-Fit-Algorithmus vorgenommen; von allen möglichen Umbrüchen eines Absatzes wird letztlich derjenige ausgewählt, der dem idealen Umbruch am nächsten kommt. Der ideale Umbruch hat zwischen allen Wörtern denselben Abstand und kommt ohne Trennungen aus. TeX geht hierbei in maximal drei Stufen vor:

1. Versuch, einen optimalen Umbruch ohne Trennungen zu erhalten.
   - ▷ Genügt das Ergebnis den Vorgaben, wird der Umbruch akzeptiert und als abgeschlossen betrachtet.
   - ▷ In der TeX-Terminologie bedeutet dies, dass die Dehnungspunkte (`\badness`) *jeder* umbrochenen Zeile den Wert von `\pretolerance=100` nicht überschreiten.
   - ▷ Wegen nicht vorhandener Trennstellen benötigt der 1. Durchgang sehr wenig Rechenzeit!
2. Versuch, einen optimalen Umbruch mit Trennungen zu erhalten.
   - ▷ Es kommen als sprachenspezifische Trennstellen die Textstellen in Frage, die TeX durch seinen Trennalgorithmus findet.

- ▷ Der Umbruch wird akzeptiert, falls alternativ
  - – die Dehnungspunkte (`\badness`) jeder umbrochenen Zeile den Wert von `\tolerance=200` nicht überschreiten und `\emergencystretch` einen positiven Wert aufweist;
  - – falls `\tolerance` den Wert `10000` hat; dann kommen zusätzlich die Werte `\hfuzz=0.1pt` und `\hbadness=1000` zum Tragen.

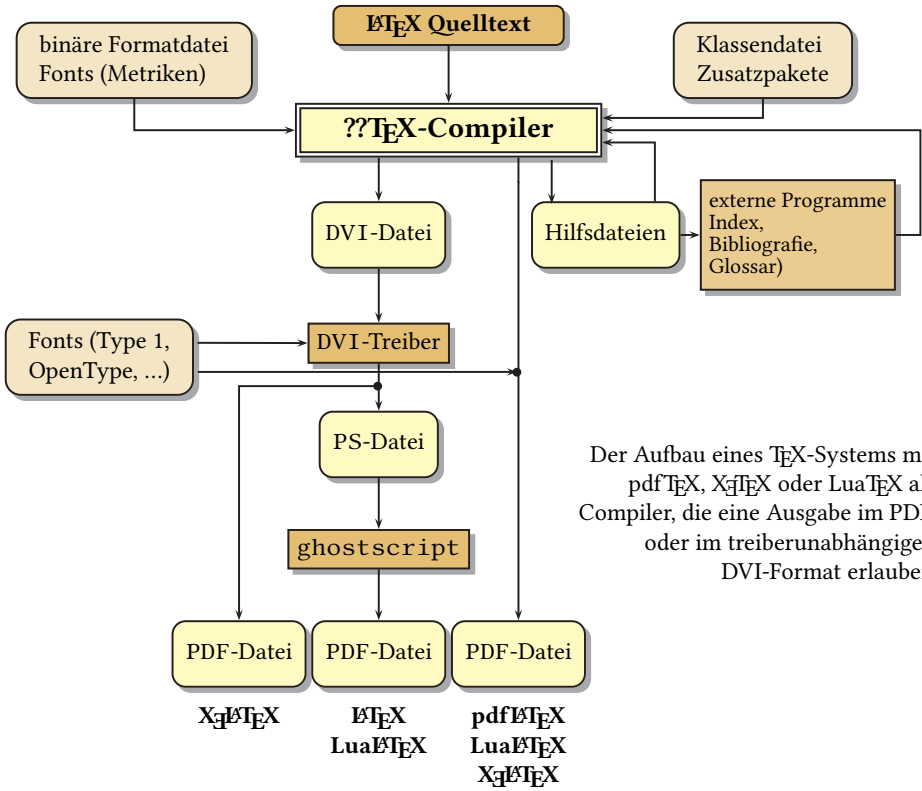
3. und letzter Versuch, durch zusätzliche Zwischenräume.
   - ▷ Der vorgegebene Wert von `\emergencystretch=9.99756pt`, der auch vom Anwender geändert werden kann, wird zur erlaubten Dehnungsmöglichkeit jeder Zeile addiert (siehe auch Abschnitt 5.9.4 auf Seite 172).
   - ▷ Durch diese Addition werden die zulässigen Dehnungspunkte jeder Zeile verringert.
   - ▷ Die abschließende Entscheidung wird mithilfe der Register `\hfuzz=0.1pt` und `\hbadness=1000` getroffen.

Hierbei wird, wie oben erwähnt, die Berechnung sofort nach der ersten oder zweiten Stufe beendet, falls der Umbruch den internen Vorgaben entspricht. Auf den sehr aufwändigen Algorithmus der einzelnen Stufen soll hier nicht weiter eingegangen werden; es wird auf die ausführliche Beschreibung in [49] verwiesen. Hier sollte nur gezeigt werden, welchen Aufwand TEX im Gegensatz zu herkömmlichen Textverarbeitungsprogrammen treibt, um einen optimierten Umbruch zu erreichen.

1.3 Das Prinzip

Es wurde bereits darauf hingewiesen, dass man sich eher als *Programmierer* denn als *Texteingeber* bei der Erstellung seines Dokumentes fühlen sollte. Dies erleichtert das Verständnis für das Prinzip bei der Texteingabe. Bei einer optimalen Situation kann man bei Auswahl einer bestimmten Dokumentenklasse erreichen, dass man lediglich die Sprachauswahl, Eingabekodierung und Schriftkodierung vorgeben muss. Sämtliche anderen Vorgaben, die das Layout und die Struktur des Dokumentes betreffen, sind bereits in der gewählten Dokumentenklasse vorgegeben, sodass der so genannte Übersetzungsvorgang ein fertig formatiertes Dokument höchster Qualität liefert.

Abbildung 1.5 auf der nächsten Seite zeigt den kompletten Ablauf eines Übersetzungsvorganges zur Erzeugung eines LATEX-Dokumentes, wie es der augenblickliche Standard ist. Der Quelltext muss wie jeder andere Quelltext eines beliebigen Programms, beispielsweise C++ oder Perl, syntaktisch korrekt sein, wenn eine fehlerfreie Ausgabe erfolgen soll. Je nach Komplexität des Dokumentes kann es notwendig sein, mehrere Durchläufe sowohl des TEX-Compilers als auch anderer externer Programme vorzunehmen. Zum Beispiel das Inhaltsverzeichnis: Beim ersten Durchlauf erzeugt der TEX-Compiler aus allen Überschriften das Inhaltsverzeichnis (Die Informationen werden in einer separaten Hilfsdatei gespeichert – Dateiendung `.aux`). Da das Inhaltsverzeichnis normalerweise am Anfang des Dokuments steht, liegt es beim ersten Start noch nicht vor. Erst beim zweiten Durchlauf wird das Inhaltsverzeichnis eingebunden. Dies kann aber dazu führen,



Der Aufbau eines TeX-Systems mit pdfTeX, XeTeX oder LuaTeX als Compiler, die eine Ausgabe im PDF- oder im treiberunabhängigen DVI-Format erlauben.

Abbildung 1.5: »Von der Quelle bis zur Mündung ...«.

dass alle nachfolgenden Seiten nach hinten verschoben werden, wenn das Inhaltsverzeichnis mehrere Seiten umfasst und so eventuelle Verweise auf Seitenzahlen nicht mehr stimmen. Somit muss ein dritter Durchlauf gestartet werden, dass somit alles komplett ist. Eine normale Textverarbeitung wie LibreOffice macht diesen Vorgang im Wesentlichen auch, nur dass man von diesen mehreren Durchläufen nichts mitbekommt und vor allem keinen Einfluss darauf hat.

Zu den externen Programmen zählen insbesondere eine Bibliografie, ein Index oder ein Glossar. Die Anforderungen an den Anwender sind hierbei gering, selbst für den Fall, dass er nicht mit einem Makefile oder einer grafischen Entwicklungsumgebung (GUI) für LaTeX arbeitet. Die gesamte Schrifteinbindung, die in der Abbildung 1.5 dargestellt ist, erfolgt vollautomatisch und muss den Anwender nur in seltenen Ausnahmefällen interessieren.

Für einfache Texte gelten nachfolgende Schritte. Verwendet man eine grafische Oberfläche für die Erstellung (siehe Kapitel 2 auf Seite 29), so kann diese einige Schritte automatisch übernehmen und so vereinfachen.

1. Texte im Editor schreiben, beispielsweise mit *Kile*: `kile meinText.tex`
2. Dokument übersetzen, beispielsweise mit pdfLaTeX: `pdflatex meinText.tex`

Es werden dabei mindestens folgende Dateien erzeugt:

`meinText.log` Enthält alle Statusmeldungen des Übersetzungsvorgangs.

`meinText.aux` Enthält unter anderem die Einträge für Querverweise.

`meinText.pdf` Das erzeugte PDF-Dokument (nach dem ersten Durchlauf ohne Inhaltsverzeichnis).

3. Erneutes Übersetzen des Dokuments: `pdflatex meinText.tex`
4. PDF-Dokument ansehen, beispielsweise mit dem PDF-Viewer *okular*: `okular meinText.pdf`

1.4 TEX, pdfTEX, XETEX, LuaTEX, ...

Der Anfänger wird mit vielen verschiedenen Begriffen überhäuft, deren genaue Bedeutung auch geübten Anwendern nicht immer klar ist. Das ursprüngliche TEX existiert im Prinzip nur noch als Kern sämtlicher Nachfolger, von denen pdf(e)TEX zur Zeit die eigentliche »Maschine« ist. Das »e« steht hier für *extended*, was sich in erweiterten Fähigkeiten des Kernprogramms TEX äußert. Da Knuth verfügt hat, dass das Original-TEX beliebig verändert werden darf, jedoch dann einen neuen Namen bekommen muss, wird man bei den aktuellen Nachfolgern oft etwas andere Namen finden, beispielsweise eTEX.

1.4.1 Ein wenig Hintergrundwissen

Das von Donald Knuth geschriebene Programm TEX stellt nur die Basisfunktionalität zur Verfügung, ist aber ansonsten genauso nutzlos wie ein C-Compiler ohne Bibliotheken. In beiden Fällen ist die Benutzung extrem umständlich. Knuth hat daher ein Makropaket (`plain.tex`) mitgeliefert, welches den Umgang mit den Basisfunktionen erheblich erleichtert. Genauso wie die Standard-Bibliotheken eines Compilers, die erst ein effizientes Programmieren ermöglichen.

Eine andere Erweiterung ist LATEX von Leslie Lamport. Nun könnte man sich vorstellen, dass man dem Anwender sagt, er solle einfach `\input plain` oder `\input latex.ltx` als erste Zeile in sein Dokument schreiben. Abgesehen davon, dass es nicht sonderlich effizient ist, wird es in der Praxis anders gemacht: Es werden dem Anwender separate Programme zur Verfügung gestellt, nämlich eTEX und LATEX, die entweder die `plain`-Makros oder die LATEX-Makros laden. Dadurch kann man LATEX-Anwendern den Eindruck vermitteln, dass LATEX ein eigenständiges Programm ist. Ein LATEX-Anwender muss prinzipiell nicht wissen, wie Dinge intern implementiert worden sind.

TEX wurde später von Hàn Thế Thành erweitert, um auch direkt eine PDF-Ausgabe erzeugen zu können. Damit kamen zwei neue Programme, pdfTEX und pdfLATEX ins Spiel, später kam XETEX von Jonathan Kew, eine auf Unicode basierende Erweiterung von TEX hinzu, welches standardmäßig eine PDF-Ausgabe erzeugt. Die neuste Entwicklung ist LuaTEX, es basiert auf pdfTEX und Unicode und hat Lua als Skriptsprache integriert. Es erzeugt ebenfalls standardmäßig eine PDF-Ausgabe. Es existieren also folgende Programme: TEX, LATEX, pdfTEX, pdfLATEX, XETEX, XELATEX, LuaTEX und LuaLATEX, wovon hier nur diejenigen von Interesse sind, die »LA« in ihrem Namen haben.