Hoe met LaTeX een boek kan worden gemaakt

Rein Smedinga

Vakgroep Informatica
Rijksuniversiteit Groningen
Postbus 800
9700 AV Groningen

rein@cs.rug.nl

Abstract

Het creëren van een boek met LaTeX is geen vanzelfsprekende bezigheid. Onderstaand het relaas van zo'n poging: het boek-in-wording *Inleiding Informatica* dat in eerste instantie als dictaat bij het bijbehorende college is geschreven en nu in een boekversie is aangeboden aan Addison-Wesley.
In onderstaand betoog wordt voorbij gegaan aan het belangrijkste onderdeel van het schrijven van een boek: het schrijven van de tekst. We zullen het hier slechts hebben over de problemen en oplossingen voor wat betreft de layout.

1 De stijl

Addison-Wesley kent een aantal regels waaraan de layout van een boek moet voldoen. Ondanks het feit dat Addison-Wesley de hoofduitgever is voor wat betreft TeX-materiaal, kent de Nederlandstalige tak (waarvoor het boek bedoeld is) geen mogelijkheid tekst langs electronische wijze aan te bieden. Nodig is dus een camera-ready manuscript met een layout overeenkomstig de eisen en regels van de uitgever .

Het boek *Inleiding Informatica* is niet mijn eerste boek bij Addison-Wesley. Voorheen verscheen *Simulatie en Implementatie*. Voor dat boek heb ik de layout al eens moeten aanpassen aan de eisen van de uitgever. Dit heb ik gedaan door uit te gaan van de LaTeX-stijl `book` en deze aan te passen. Belangrijkste aanpassingen zijn de gewijzigde hoofdstuktitels (cursief, nummer erboven, rechts aangelijnd, zonder het woord hoofdstuk), geen extra lege regels tussen opsommingen, terwijl alle indentaties (nieuwe paragraaf, items in `itemize` en `enumerate`, formules) gelijk zijn. Verder dienden de sectie-titels te worden aangepast (font, ruimte eromheen e.d.), de titels van figuren links aangelijnd te zijn (niet gecentreerd) en waren bepaalde waarden opgegeven voor tekstbreedte en -hoogte. Eén en ander bleek redelijk eenvoudig in de files `book.sty` en `bk12.sty` aan te passen. Zo ontstonden `awbook.sty`, `awbk12.sty` en later ook `awbk11.sty` voor eigen gebruik bij dictaten.

Naast `awbook` zijn nog een aantal aanvullende `sty`-files vereist. We noemen `fleqn` (om formules links te beginnen in plaats van gecentreerd), `which` (zie verderop), `pict` (de file `pict.sty` bevat een groot aantal macros voor het tekenen van eindige automaten. Een aantal daarvan zijn ook nuttig voor tekeningen in dit boek), `idxans` (zie verderop), `inlmacros` (alle macros, die voor het boek nodig zijn, zoals vetgedrukte **en**, stelling en opgave-omgevingen e.d.) en `program2` (de zelfontworpen programma-omgeving waarmee PASCAL-programma's elegant kunnen worden gelayout, zie verderop).

2 De indeling

Zoals Lamport al aangeeft in zijn boek over LaTeX is het verstandig een boek op te delen in hoofdstukken en per hoofdstuk een file aan te maken. Hiervoor geschikt zijn de macros `\include` en `\includeonly`. De laatste heeft als parameter de naam van de file die verwerkt moet worden. Van alle ge-`\include`-de files wordt wel de betreffende `.aux` file gelezen, maar verwerking van de tekst vindt alleen plaats als de filenaam ook als parameter in `\includeonly` meegegeven is. Zo kan inderdaad netjes hoofdstuk voor hoofdstuk worden afgewerkt zonder problemen te hebben met paginanummering, cross-referenties e.d.

Het boek is dan ook onderverdeeld in een hoofdfile `dictaat.tex` (Het boek bestond al als dictaat voordat het een boek werd!) waarin de nodige macro-files worden aangeroepen en verder slechts aanroepen staan van `\include` (voor elk hoofdstuk één). De tekst van de hoofdstukken is vervolgens terug te vinden in de files `ch00.tex` tot en met `ch18.tex`. Aangezien later hoofdstukken zijn toegevoegd bestaan ook de files `ch11b.tex` en `ch14b.tex`. Ook voorwoord, inhoud, index, e.d. hebben een eigen file, maar daarover later meer.

3 \which

Voor eerdere boeken (en artikelen met hoofdstukken) is al eens de macro `\which` ontworpen, die in grote trekken overeenkomt met de standaard macro `\includeonly` met dat verschil, dat het argument van de laatste interactief kan worden opgegeven. Bovendien worden alle files verwerkt, indien de vraag wordt beantwoord met een return. De macro `\which` is alsvolgt gedefinieerd:

```
\def\which{\typeout{%
Give file(s) to be processed.}
\typein[\file@name]{%
(without .tex, separated by commas,
return for all files)}
\ifx\@empty\file@name \else
\includeonly{\file@name} \fi}
```

De `\ifx` vergelijkt de ingevoerde filenaam met de lege macro en roept alleen `includeonly` aan indien de filenaam niet leeg is. Is de naam wel leeg, dan volgt geen aanroep van `\includeonly` en worden dus alle files verwerkt. `\include` heeft dan ongeveer dezelfde betekenis als `\input`.

4 \include

De macro `\include` heeft als grote bezwaar, dat bijvoorbeeld de index steeds terechtkomt in de file `dictaat.idx`. Als we een hoofdstuk verwerkt hebben en aan een nieuw hoofdstuk beginnen worden de oude gegevens in `dictaat.idx` overschreven door de gegevens uit het nieuwe hoofdstuk.

Om dit op te lossen is de macro `\include` vervangen door de macro `\Include` die per hoofdstukfile een `.idx` file aanmaakt. Bij verwerking van een volgend hoofdstuk blijven de gegevens van het vorige hoofdstuk dus netjes bewaard. Voor de index betekent dat dan dat (afgezien van sorteren e.d. van de index-items) we alleen nog maar de files `ch00.idx` tot en met `ch18.idx` achterelkaar behoeven te zetten en te verwerken. Bij gebruik van de oude `\include` moet de laatste keer het hele boek worden ge-LaTeX-ed om een lijst met index-items te verkrijgen voor het gehele boek.

Het boek bevat verder veel opgaven en het leek handig de antwoorden van de opgaven direct in de tekst erbij te kunnen typen en LaTeX ervoor te laten zorgen dat deze naar een aparte file worden geschreven om later in één keer weer te worden ingelezen. We hebben hiervoor gebruik gemaakt van de macro `\answer` zoals beschreven in het TeXbook van Knuth:

```
\newwrite\ans
\immediate\openout\ans=\jobname.ans
\outer\def\answer{\immediate\write\ans{}
\immediate\write\ans{\string\antwoord{\theopg}}
\copytoend}
\def\copytoend{\begingroup\setupcopy\copyans}
\def\setupcopy{\catcode`\\=12 \catcode`\{=12
\catcode`\}=12 \catcode`\$=12 \catcode`\&=12
\catcode`\#=12 \catcode`\%=12 \catcode`\~=12
\catcode`\_=12 \catcode`\~=12 \catcode`\ =12
\catcode`\ =12 \obeylines}
{\obeylines\gdef\copyans#1
{\def\next{#1}%
\ifx\next\empty\let\next=\endgroup%
\else\immediate\write\ans{\next}
\let\next=\copyans\fi\next}}
```

Voordeel van deze methode is verder dat de nummers van de opgaven (deze staan in een counter met de naam `opg`) direct beschikbaar zijn (als `\theopg`) en niet met behulp van `\label` en `\ref` opgeslagen behoeven te worden (bovendien spaart dit geheugenruimte en verkleint het de kans op de mededeling dat de capaciteit van TeX overgeschreden is).

Gebruik van de macro `\answer` eist de nodige zorgvuldigheid: alle tekst tot de eerstvolgende lege regel wordt naar de antwoordenfile weggeschreven.

Evenals bij de index willen we graag per hoofdstuk een antwoordenfile (met extentie `.ans`). De macro `\Include` ziet er dan alsvolgt uit:

```
\def\Include#1{\clearpage
\if@filesw \immediate \write \@mainaux
{\string \@input {#1.aux}}\fi
\@tempswatrue \if@partsw \@tempswafalse
\def \@tempb {#1}\@for
\@tempa :=\@partlist \do {\ifx \@tempa
\@tempb \@tempswatrue \fi }\fi
\if@tempswa \if@filesw
\let \@auxout =\@partaux
\immediate \openout \@partaux #1.aux
\immediate \write \@partaux {\relax }\fi
\immediate \closeout\@indexfile
\immediate \openout \@indexfile #1.idx
\immediate \closeout\ans
\immediate \openout \ans #1.ans
\@input {#1.tex}\clearpage
\@writeckpt {#1}\if@filesw
\immediate \closeout \@partaux \fi
\let \@auxout =\@mainaux \else
\@nameuse {cp@#1}\fi }
```

Het verwerken van één hoofstuk heeft nu tot gevolg dat voor dat hoofdstuk een file onstaat met de bijbehorende index-items en met de bij de opgaven uit dat hoofdstuk behorende antwoorden.

De macros `\Include` en `\answer` zijn uiteindelijk samen gezet in de file `idxans.sty`, die nu binnen onze vakgroep voor algemeen gebruik beschikbaar is.

Voor het afzonderlijk bekijken van de antwoorden van één hoofdstuk (en de index) is er een tweede mainfile gemaakt: `answer.tex`. Deze werkt weer gewoon met `\include` en geeft de mogelijkheid (m.b.v.

\which) de antwoorden en de index-items per hoofdstuk te verwerken. Vooral aan het begin is dit handig omdat dan niet alle antwoorden ge-LaTeX-ed behoeven te worden en steeds locaal per hoofdstuk kan worden gewerkt. De index-items worden in volgorde van optreden in de tekst genoteerd. Hiervoor is in answer.tex een macro \itementry gedefinieerd (alle indexitems komen standaard voorafgegaan door deze macro-aanroep in de .idx-file, dit doet LaTeX) die een item in de indexlijst oplevert. Deze indexverwerking gaat dus buiten makeindex (zie verderop) om.

5 De index

Voor het maken van de index kon gelukkig op het laatste moment gebruik worden gemaakt van het programma makeindex. Deze heeft een aantal handige faciliteiten, zoals het in de index opnemen van pagina-ranges waarin een term wordt uitgelegd:

```
\index{term|(}          % op beginpositie
\index{term|)}          % op eindpositie
```

het gebruiken van subitems en subsubitems:

```
\index{term!subterm!subsubterm}
```

het in de index plaatsen van verwijzigen naar een ander woord:

```
\index{term|see{other term}}
```

en het in andere stijlen (bold, cursief, in mathmode e.d.) opnemen van woorden zonder dat dit de sortering verstoort:

```
\index{en@{\bf en}}
```

Belangrijk is overigens al dat het programma de indexitems sorteert en gelijke voorkomens bijelkaar veegt. Een heel plezierig programma. In ons geval is nu slechts nodig alle .idx files in één nieuwe file te zetten (met behulp van het UNIX commando cat bijvoorbeeld), makeindex hierop los te laten en het uiteindelijke resultaat nog eens door LaTeX te halen. Lange tijd is overigens besteed aan het opsporen van fouten in de index (vooral het terugzoeken van de bijbehorende macro-aanroep in de tekst) en het consistent krijgen van de index (geen woorden soms als item en soms als subitem opnemen, bijvoorbeeld). Klein probleem was verder nog dat de uiteindelijke file begint met \begin{theindex}. In LaTeX wordt standaard een index niet in de inhoud opgenomen. Dit kan wel door na \begin{theindex} een extra macro aan te roepen(nl. \addcontentsline), maar dat is in ons geval weer onmogelijk omdat we dan de file die makeindex oplevert steeds weer opnieuw eerst zelf zouden moeten editen. Een oplossing is een herdefinitie van de macro \theindex die uiteindelijk door \begin{theindex} wordt aangeroepen:

```
\let\oldindex\theindex
\def\theindex{\oldindex
\addcontentsline{toc}{chapter}{Index}}
\input{index.tex}
```

waarna de door makeindex geproduceerde file index.tex direct met het gewenste effect kan worden ingelezen. Voorts diende de verwijzing in de index aangepast te worden aan het Nederlands:

```
\def\see#1#2{{\it zie} #1}
       % #2 is pagenumber, to be omitted
```

6 Programma-omgeving

Oorspronkelijk stonden de grotere programma's in een figure-omgeving (één van de twee mogelijkheden zogenaamde zwevende tekst te krijgen. De ander is de table-omgeving), maar aardiger is het een speciale programma-omgeving te hebben: een zwevende tekst, waarin een programma gezet kan worden, die in de bijbehorende titel (de caption) ook netjes programma heet en niet figuur of zo. Hierin is LaTeX heel netjes, slechts een paar regels waren nodig om dit opgelost te krijgen:

```
\def\listofprogrammes{\@restonecolfalse\if@twocolumn\@restonecoltrue\onecolumn
 \fi\chapter*{\@PLijst}\@mrkr{\@PLijst}%
\@starttoc{lop}\if@restonecol\twocolumn
 \fi}
\def\l@programme{\@dottedtocline{1}{1.5em}{2.3em}}
\newcounter{programme}[chapter]  % new counter
\def\theprogramme{\thechapter.\@arabic\c@programme} % counter layout
\def\fps@programme{tbp}          %
\def\ftype@programme{3}          % figure has type 1, table type 2
\def\ext@programme{lop}          % listing in file .lop
\def\fnum@programme{\@Programme\ \theprogramme} % caption number layout
\def\programme{\@float{programme}} % the real definition
\let\endprogramme\end@float
\@namedef{programme*}{\@dblfloat{programme}} %idem for twocolumns
\@namedef{endprogramme*}{\end@dblfloat}
\def\@PLijst{Lijst van programma's} % title for list-chapter
```

Overigens moest hier gebruik worden gemaakt van de naam programme omdat program al een omgevingsnaam is (zie verderop).

7 Programma-layout

Er zijn in feite twee methoden om een (PASCAL-)-programma(fragment) in de tekst op te nemen: een eenvoudige en een mooie. De eenvoudige manier is gebruik te maken van de `verbatim`-omgeving. De programma's verschijnen dan in teletype-font en letterlijk zoals ze zijn ingetikt. De mooie manier is gebruik te maken van vetgedrukte gereserveerde woorden, in math-mode geplaatste statements, in teletype geplaatste strings en alles volgens een strakke, van te voren vastgelegde layout. Mijn voorkeur ging al direct uit naar de tweede manier.

In eerste instantie verzon ik een groot aantal macro's om de gereserveerde woorden in boldface te krijgen zonder iedere keer dingen hoeven in te tikken zoals `{\bf program}`. Zo ontstonden de macro's `\PROGRAM` e.d. Verder maakte ik gebruik van de `tabbing`-omgeving om een nette layout te krijgen. Later bedacht ik, dat het verkrijgen van een nette layout wellicht meer geautomatiseerd kon door de `tabbing`-macros op te nemen in de gemaakte macros.

Aangezien een programma uit een aantal levels bestaat (elke procedure-aanroep creeërt een nieuw level en bij beëindiging van een procedure-body wordt weer teruggegaan naar het vorige level) was iets nodig als een stapelmechanisme om oude tab-settings te bewaren. Dit mechanisme is in TeX standaard aanwezig in de vorm van groups. Bij het verlaten van een binnengroup worden de waarden en definities van de omvattende group teruggezet. Hier heb ik dan ook driftig gebruik van gemaakt. Verder kan met behulp van \+ en \− worden geregeld dan de eerste voorkomens van \> in een regel achterwege gelaten kunnen worden.

Op de een of andere manier heb ik gekozen voor een layout met de volgende eigenschappen:

1. locale declaraties en de body van een procedure zijn van een dieper niveau dan de procedureheading.
2. de programma-heading, de globale procedure-headings en het hoofdprogramma behoren tot het bovenste niveau.
3. de formele specificatie van een procedure (of functie) behoort tot hetzelfde niveau als de heading.
4. een dieper niveau springt in.
5. bij het herhalingsstatement staat de **do** onder de **while** indien het herhalingstatement een compound statement is. Evenzo voor **if**, **then** en **else** en vergelijkbare statements.
6. puntkomma's tussen statements in een compound-statement staan niet achteraan de regel, maar vooraan de volgende, onder de **b** van **begin**. De **e** van de afsluitende **end** staat hier weer onder.
7. declaraties over meerdere regels springen op volgende regels in.

Uiteindelijk resulteerde dit in een de verzameling macro's die is terug te vinden in de stijl `program2`.[1] Om niet onnodig $-tekens te hoeven tikken heb ik naast `tabbing`-omgeving een `mathtabbing`-omgeving gemaakt (`tabbing` in mathmode).

De uiteindelijke programma-stijl wordt op dit moment ook door andere leden van de vakgroep gebruikt. Een uitgebreide handleiding hiervoor is beschikbaar onder de naam `program.tex`.

Als illustratie een aantal fragmenten uit de `sty`-file. Het stapelmechanisme is geïmplementeerd door de macros `\@push` en `\@pop`. De teller `\@tabs` geeft het aantal extra tabposities in de nieuwe omgeving aan. De teller `\@etabs` geeft het aantal extra terug te springen tabposities aan bij verlaten van het level. Bijvoorbeeld: een **end** komt te staan onder de **begin**, dat is in het te verlaten level `\@tabs` tabposities terug. In de regel na deze **end** moet er nogeens `\@etabs` posities worden teruggesprongen (het compound statement sprong naar alle waarschijnlijkheid al een aantal tabposities in). De macro's `\@stopf` en `\@contf` komen uit de `tabbing`-omgeving. De macro `\@test` is een debug-faciliteit en drukt de waarden van een aantal tellers af.

```
% \@pushXX creates following level and defines \@tabs to be XX
\def\@push#1{%\@test@{push}
\@stopf\global\advance\@level by1\relax \begingroup
 \@tabs#1\relax \@etabs=0 \relax\@contf}
% \@addXX adds the value XX to \@tabs
\def\@add#1{\@stopf\advance\@tabs by#1\relax\@contf}
%
%\@extraXX sets extra \- to be done by reentering this level
\def\@extra#1{%\@test@{extra}
\@stopf\advance\@etabs by#1\relax\@contf}
%
% \@pop returns one level and does the needed number of \- and \<
% it does \-\< as many times as mentioned in \@tabs and \- as mentioned in
% \@etabs
\def\@pop{%\@test@{pop}
\ifnum\@level<0\@warning{Negative level in program environment}
\else\@stopf\global\advance\@level by-1 \global\@tmp=\@tabs \@contf%
```

[1] Er is ook een stijl `program` waarin de mathematische mode ontbreekt.

```
\@ifatmargin
 \ifcase\@tmp\or\<\-\or\<\<\-\-\or\<\<\<\-\-\-\or\<\<\<\<\-\-\-\-\fi
\else \sp@c % mathspace
 \ifcase\@tmp\or\-\or\-\-\or\-\-\-\or\-\-\-\-\fi
\fi \@stopf
\endgroup  \global\@tmp=\@etabs \@etabs=0 \@contf%
\ifcase\@tmp\or\-\or\-\-\or\-\-\-\or\-\-\-\-\fi \fi }
```

Als voorbeeld het gebruik van `\@pop` en `\@push` voor de **begin** en **end**-symbolen:

```
\def\@plusb{\@stopf\advance\@Begins by1\@contf}
\def\@minusb{\@stopf\advance\@Begins by-1\@contf}
\def\@begin@#1{\enddecl\endheading#1\=\+\@push1\@plusb}
\def\@end@#1{\@minusb
          \ifrecordcase\@pop#1\@pop\@stopf\recordcasefalse\@contf
          \else\@pop#1\fi\@stopf\recordfalse\@contf
          \ifbody\ifnum\@Begins=0\ifnum\@level>0
             \@stopf\bodyfalse\@contf \@pop \fi\fi\fi}
\def\BEGIN{\@begin@{\mbox{\bf begin }}}
\def\END{\@end@{\mbox{\bf end }}}
```

Vooral de `\END` is lastig. Een **end** kan zowel de afsluiter zijn van een **begin** (waarbij het verschil maakt of dit een gewoon compound statement is of een body, in het laatste geval sluit de **end** tevens de procedure declaratie af), als van een **record** (eventueel een variant record), als van een **loop** (in ModPas). Zonodig is van conditionals gebruik gemaakt om het onderscheid te kunnen maken.

Andere overwonnen moeilijkheden in deze omgeving zijn de indendatie van procedurebodies en locale declaraties, terwijl de regels tussen de heading en de locale declaraties gebruikt kan worden voor een formele specificatie en niet ingesprongen worden, de verschillende betekenis van bijvoorbeeld `\VAR` als aankondiging van een variabele-declaratie en als aankondiging van een locale var-parameter (in het laatste geval dienen geen tabsettings plaats te vinden, in het eerste (meestal) wel), de juiste indentatie voor compoundstatements achter een **do** van een herhalingsstatement, achter een **then**, enzovoorts.

Naast de macros zoals `\PROGRAM` bestaan er ook macro's `\Program` die een gereserveerd woord vetgedrukt afdrukken: **program**-tekst krijg je dan door

```
\Program-tekst
```

te tikken.

8 Ergenissen

De grootste ergenis bij het gebruik van LATEX is het veelvuldig optreden van de mededeling

```
TeX capacity exceeded
```

als gevolg van het gebruik van de `picture`-omgeving in combinatie met een redelijke hoeveelheid cross-referenties. Een aantal keren heb ik de mededeling (en het afbreken van de verwerking) kunnen voorkomen door een aantal `aux`-files tijdelijk weg te halen (de cross-referentie-lijst werd dan minder lang) en een aantal keren door de plaatjes uit de tekst te halen (d.i. nog slechts een lege `picture`-omgeving laten staan) en in een aparte file het werkelijke plaatje onder te brengen. De bijbehorende plaatjes moeten dan wel met de hand worden ingeplakt.

Helaas wordt in LATEX voor elk label een macro-definitie gemaakt en deze wordt op dezelfde plaats opgeslagen als de nog niet ge-`\output`-te ingelezen tekst. Zeker bij plaatjes in een zwevende tekst omgeving loop je dan het risico de maximale hoeveelheid te gebruiken ruimte te overschrijden (overigens een kwestie van het integer-bereik van de gebruikte PASCAL-implementatie, is mij eens verteld).

9 Conclusie

Wie een boek schrijft en het manuscript camera-ready aanlevert haalt zich dus veel werk op de hals. Gebruik van TEX (en vooral LATEX) heeft dit werk sterk vereenvoudigd (veel is al mogelijk of eenvoudig te maken en met behulp van `makeindex` is nog nooit zo snel en volledig correct een index gemaakt) en ook weer sterk bemoeilijkt (omdat het zo mooi kan, wil je het ook zo mooi mogelijk, wordt je verschrikkelijk precies en ga je alle mogelijkheden benutten).

Overigens, een ieder die iets van bovenstaande wil gebruiken (of gedetailleerder wil bekijken) kan mij een email sturen.

LADIES and LATEX

Th. A. Jurriens

Kapteyn Astronomical Institute,
University of Groningen,
P.O. Box 800,
9700 AV Groningen

taj@rugr86.rug.nl

Abstract

This paper describes the use of LATEX by non-astronomers at the Kapteyn Institute. Although the general opinion is that it is too complicated for non-scientist, experience in Groningen proves otherwise. If you provide good tools or styles everybody is capable of using LATEX. Also, the astronomers took over tricks from the ladies, simplifying their lives, too.

1 Introduction

Just before my first LATEX course, given to the secretaries at the Kapteyn Institute, [1] I had a discussion with colleagues at the Computing Center. The subject of discussion was the title of this paper "Ladies and LATEX" – according to my colleagues of the Computing Center it wasn't the right combination. But the experience gained from five years intensive LATEX use shows a complete different result: the secretaries of our Institute are now experienced LATEX users and in most cases they are able to solve their own LATEX problems. So this paper isn't intended to be unfriendly towards ladies.

2 History

Before introducing LATEX as the **document preparation tool** the ladies were used to a "what you see is what you get" editor, working on a system called A.E.S. It was a huge computer with two workstations and one big noisy printer. Learning wasn't easy at all and the system wasn't open to the world. More astronomers started to type their own texts using TEX: the quality of these were much better than that of the texts produced by the ladies. So it was clear: the A.E.S. system was out of date.

3 Why LATEX?

In 1987 I started to use TEX. I wrote my own macros and concluded that in a way I was writing my own kind of LATEX. I was forced to explore LATEX upon request of my institute, which was introducing LATEX as the **document preparation tool**. In 1987 only Word Perfect 4.0 was available and wasn't capable of producing formulas and nicely lined-up tables – always in need in astronomy. So Word Perfect wasn't the solution for our ladies. Different reasons also existed for not selecting Chi-Writer, at that time the favourite editor of our colleagues at the Deparment of Mathematics.

In order to train seven secretaries in the use of LATEX I set up a three-day course – which since the first "performance" in Erm has been used several times[2] , held in the middle of nowhere to guarantee that we weren't disturbed by daily work, phone calls, etc. The result of this training was a cook-book [2], full of examples, which is still in use. I used Urban's manual [4] as the main text for the course. Table 1 shows the set-up of the course. Chapters refer to Urban's manual [4].

During that time I also used LATEX to edit my own books [3]. The editing of an astronomical yearbook proved to me that LATEX is 99% powerful enough.

[1] The Institute incorperates the Kapteyn Laboratory of the University of Groningen, the Kapteyn Observatory in Roden and the Laboratory for Space Research

[2] 1990: Department of Mathematics, 1991: Department of Mathematics Groningen University, Computing Center Novosibirsk USSR

lesson 1	Chapter 1 & 2
	Introduction & Getting Started
2	Chapter 3
	Control Sequences
3	Chapter 4
	begin and end environments
4	Chapter 5, partly Chapter 6
	Putting It Together
5	Chapter 6,7 & 8
	Tables and Figures,
	Cross-References,
	Equation Formatting
6	Chapter 8
	Equation Formatting

Table 1: Course setup, each part is 3 hours

4 Restrictions

The ladies had several technical requests concerning the possibilities of the text editing system/environment:

1. To have their own private printer so that they could print confidential papers or letters and avoid the queue of the general printer.
2. Printing in landscape mode.
3. The possibility of previewing.
4. When necessary, to be independent of the local area network.
5. Easy to learn.

Besides these technical issues, there were more restrictions involved with the type of work of our secretaries: making mailing labels, typing letters, manipulating databases, producing forms and preparing manuscripts for publishers. To solve these problems I created several tools which I will describe in this paper. The problem of preparing manuscripts was solved by the publishers: more and more publishers are adapting LaTeX or TeX as the standard document preparation tool. According to our "ladies" the styles provided are easy to use.

5 Tools

5.1 Letters

To easy life, the use of LaTeX, we created some tools and adapted several style files. For example the plain `letter.sty` was not good enough for our purposes. About 75% of our outgoing mail are letters so we adapted the letter style file to suit our needs and also included the symbols of our letterhead (see figure 1). Both are adapted from a description for a Kyocera laserprinter. It is converted to a 300 dpi pxl-file and is created without a `MF` - file. The rules of our University regarding letters require certain characteristics of a letter, like the address, date, etc., to be at a fixed position on the page. To obtain this result a picture environment is used.

Figure 1: The two elements of the style of the University.

Our `kapteyn.sty` letter style knows two languages: Dutch and English. With one switch we can change the language of the fixed elements like the date, the name of the Institute, etc (see table 2). An environment is also supplied for preparing telefaxes. With this environment, it is possible to count or give the number of pages. The output is in 12pt by default to avoid reading problems at the other end of the line.

English	Dutch
Our no.:	Ons nr.:
Re:	Betreffende:
cc:	I.a.a.
Encl:	Bijlage(n):
Date:	Datum
Our ref.:	Ons kenmerk
Your ref.	Uw kenmerk:
Subject	Onderwerp

Table 2: Some language-dependent elements in our letter style.

In the current working environment – a SUN platform – we are using a script to make letters. It prompts for the language and then chooses the right template for doing the job. So the secretary must simply type the letter. Of course, the scripts are used by non-secretaries too. Thus all outgoing mail now has the same face to the world. Users are able to change the phone-number and email address in the letterhead.

5.2 Mailing-labels

To create mailing-labels a three-column plain-TeX macro, available on request, was adapted for our needs. Before finding the good sizes it was necessary to play around. We use labels in use for copying machines. The adapted macro prompts a file containing the adresses. Each adress is separated by `\next`. If the file contains only one adress it will create 24 labels, the maximum of one sheet of labels. If the adress is to big it is split over two labels with a • sign to notify the person who is really doing the job of mailing.

Figure 2: The language-varying elements of our letterhead: on the left a Dutch version, on the right, the English. To be in line with official rules of the University, PostScript-fontsare used instead of the default CMR fonts.

5.3 Databases

Currently we use Dbase for keeping track of all publications made by the astronomers at our Institute and also information about our students. In the first case we mainly use the information for our Annual Report: an overview of research and all publications. Some Dbase-programms are used to separate between publications in journals, contributions in conference – proceedings, thesis etc.

6 Publications

Every year our Institute produces the earlier mentioned Annual Report and the "studiegids" a guide for students. For the purpose a `bk9pt.sty` file was created in combination with a `A5.sty`. Both publications are easy-made. To improve output quality, we use an adapted DVI2PS driver, for 600 dpi. At our University printing house we have access to a 600 dpi printer (Varitype).

7 Conclusion

LaDIES and LaTeX are a perfect couple, in other words: everybody can use it!!

References

[1] Leslie Lamport, *LaTeX user's guide and reference manual*, Addison-Wesley, Reading, Mass., 1986.

[2] Theo Jurriens et. al., *LaTeX kookboek*, 1988.

[3] Theo Jurriens, *Hemelwijzer 1989*, De Feniks Pers, Meppel, 1988.

[4] Michael Urban, *An Introduction to LaTeX* TeX Users Group, P.O. Box 9506, Providence, RI 02940, U.S.A, 1986.

Two Sides of the Fence[1]

Nico Poppelier

Elsevier Science Publishers
Academic Publishing Division, R&D Department
Sara Burgerhartstraat 25
1055 KV Amsterdam
the Netherlands

n.poppelier@elsevier.nl

Abstract

The purpose of this talk is to give an overview of the four days of the twelfth annual TUG meeting; it is an attempt to show that the different streams in the programme of the meeting are connected, that they are part of a whole.
Also, I make some comments and observations regarding the current status and the future of TEX, and the future of publishing in general.

1 Introduction

In his book *Zen Buddhism* [5], Christmas Humphreys writes:

> How then, does it work, this faculty of the mind [the intellect] which men so highly prize and far too lightly claim to be infallible? The answer is, by the interaction of the opposites.

The purpose of this talk is to give an overview of the four days of this conference, and I will use pairs of opposites to guide me through it.

If you talk about pairs of opposites, you also talk, implicitly, about a fence, a boundary between the two opposites. And if you consider any of these fences you can ask yourself: do we make an opening in the fence, i.e. make a pragmatic decision in order to bridge the gap, to integrate seemingly irreconcilable views? Or will we remain passive, will we stay 'sitting on the fence', i.e. not decide anything? There is of course a third possibility, namely that the fence is there for a real purpose.

I hope that this conference will result in gates through the various fences I will discuss.

2 Dichotomies

The first pair of opposites came into my mind very quickly: the TEX-using author vs. the TEX-accepting publisher. From the TEX files we've received so far at Elsevier Science Publishers I've gotten the impression that the average TEX-using author wants as much freedom as possible to typeset the text, the tables, the math and the figures. He/she wants to use TEX in any possible imaginable way and, according to TEX experts at a few physics institutes, spends sometimes up to 50% of the total time for the article or book on its presentation.

Suppose he has to deal with publisher X, who has a TEX macro package plus instructions to authors. Then maybe the author isn't very happy with it, since it limits him in his creativity and furthermore, since he has to deal with many publishers, he has to figure out a way of dealing with these different macro packages and instructions. A very likely solution is that he just ignores them all!

The publisher who accepts TEX has a slightly different point of view. Of course, on the one hand, a publisher wants to be as friendly as possible to an author and accept his compuscript. But, on the other hand, a publisher wants to convert the TEX compuscript into a printed book or journal paper in the shortest time possible with a minimal amount of effort.

[1] Keynote presentation at TUG91 meeting in Dedham, USA; to be published in TUGboat, © 1991, TEX Users Group.

There are several constraints to be met in this publication process: the house-style for the particular journal or book series, the quality of the publication (language, layout), the time it takes to publish the article or book, and the cost of all this. Most publishers are commercial firms, not philanthropic institutions, so cost efficiency is an important criterion. In most cases, the publisher would really like to see authors following the instructions.

How do you solve this dilemma? A compromise might be to agree upon a certain standard or set of standards between various publishers. In our company, we think that we will not be able to handle TeX compuscripts efficiently if we accept all varieties of TeX, especially because the material ranges from very simple to very complex with lots of math and tables. Efficiency is particularly important for journals, where you have a steady flow of material, a fixed house-style and a routine way of working.

Our choice is: one variety of TeX, namely LaTeX. For book and proceedings projects this preference is somewhat less strong, although making a book ready for publication, in a house-style or in the style of a particular book series, complete with a table of contents and an index, is easier if the book was prepared with LaTeX—and the author has used LaTeX well!—than if it was prepared with plain TeX.

Besides the problems just mentioned, there are several other matters you have to solve anyway, regardless of whether you use plain TeX, LaTeX or, say, $\Phi\Upsilon\Sigma$TeX:

- complex tables
- page layout
- font selection (other fonts than Computer Modern)
- illustrations in PostScript or other format

So now I've come to my second pair of opposites, one that will be addressed by several speakers this week: TeX versus LaTeX.

The key concept of LaTeX is, as you of course know, the concept of logical design: an author writes his text in terms of abstract building blocks, in terms of the logical structure of the text. Content and layout are decoupled as much as possible. The visual structure is derived from the logical structure, and is specified in the document style.

As I said earlier, some authors appear to spend large amounts of time on the presentation of a paper that is submitted for publication in a journal: they write sets of macros ranging in size from one screen to many hundreds of lines, use any font they can find in all sorts of combinations, etcetera. This strikes me as odd for two reasons: (i) an author's main concern should be the *contents* of the article or book, and (ii) the presentation the author chooses will almost always be changed by the publisher anyway, whether he submits the material on paper, on a diskette or via electronic mail.

We have found that the LaTeX-way-of-working is fine for both journals and books: document styles have been written for about ten journals and several books. The difference between conventionally typeset material and material produced from author-prepared LaTeX files can only be seen by a well-trained eye.

Now of course, there is much more to this type of electronic publishing than just changing the document style: a technical editor has to look at spelling, punctuation, language in general, notation, the appearance of mathematical formulas in text and in displays, the layout of tables, the page layout, spacing, hyphenation, . . . a lot of work, often difficult work. The combination of usual copy-editing with TeX requires skilled technical editors and a certain routine way of handling TeX.

But TeX is not the only document preparation publishers have to deal with. And so now I come to my next pair of opposites: TeX vs. non-TeX, or TeX versus the rest of the desktop-publishing world.

If we asked scientists who publish in one of our more than 600 journals whether they use a computer to write their articles and if so, what word processor they use, we would find enormous variety in their answers. In physics and mathematics, TeX is used by the majority of authors, but even there you find a significant number of authors who use `troff/eqn`, ChiWriter, Word or various Macintosh word processing programs.

In other scientific disciplines, TeX is used by only a few people—if at all! What I personally find most interesting is the many ways TeX is used, not by mathematicians and physicists, but by people working in, say, linguistics, humanities. My next pair of opposites.

Often there is no alternative but TeX for producing texts in languages that use non-Latin alphabets or the Latin alphabet with diacritical marks. With TeX you can produce remarkable, often beautiful results, after you have solved dozens of problems that others, who use TeX for texts written in English, with a lot of math and tables, have never thought of. I am fascinated by the work on

- hyphenation of other languages than English
- right-to-left text with TeX: Hebrew and Arabic
- diacritical marks and other embellishments: Hebrew, Vietnamese
- wonderful fonts: Greek, Hebrew, Arabic, Old German, Ethiopic, Korean *hangul*, Japanese *kana*, Chinese *kanji* or *hanji*, and the many languages of the Indian sub-continent
- vertical typesetting: Japanese and Chinese

and I hope to see a lot of these types of TeX applications during this conference. I think that, in principle, TeX has great potential as a text composition system for authors in *all* scientific disciplines and in *all* languages. But, I said 'in principle'—I will come back to that later.

Coming back to the observation that TeX is not the only software: when a publisher sees that he also receives papers prepared in Word and ChiWriter, what does he do with them? Does he handle them in the old-fashioned way, that is re-type the whole thing and introduce lots of typos, so that the author has to read the stuff for the umpteenth time? Or should the publisher convert it to one of the professional typesetting systems he uses? Or convert it to TEX, since there are several of these conversions available: WordPerfect to TEX, ChiWriter to TEX, ...

I think conversion will become important or is already becoming more and more important. Conversion of information from one format into another, from an author's word processor X to a publisher's typesetting system Y. Now suppose authors use M different word processors and that publishers uses N different typesetting systems: does this mean we have to wait for the development of $M \cdot N$ different conversions? This does not appear to be a feasible solution. Conversion, or translation, via an intermediate language, a standard exchange language for text, would require only $M + N$ different conversions, much less!

As most of you know, such an intermediate language already exists: SGML [6, 2, 4]. Aha, you might think: the fourth pair of opposites. Well, yes and no. Yes, in the sense that many people think that TEX and SGML are two alternatives for one and the same purpose. No, in the sense that I do not agree with this: I do not believe that SGML and TEX form a pair of opposites and I would like to explain why I think this is the case.

SGML is not a typesetting language, but an abstract language, or more precise: a meta-language. Just as you can define the computer programming languages Pascal and Modula-2 in BNF (Backus-Naur form), another example of a meta-language, you can define typesetting languages in SGML.

In SGML, there exists something that is called the *document type definition*. A document type definition (DTD) is a description of a class of documents. You describe a document instance, a document that is representative for a certain class of documents, say *book*, as a hierarchy of building blocks. To give an example:

```
book    = front_matter body back_matter
body    = chapter+
chapter = chapter_heading, paragraph?,
          section*
...
```

all the way down to the basic building blocks: paragraphs of text, mathematical formulas, ... This defines the contents of the book in terms of logical entities: you might call it 'object-oriented writing of a document'.

An alternative is to describe the visual structure of a document, which can also be regarded as a hierarchy of building blocks.

```
book        = pages+
page        = header_block text_block
              footer_block
text-block = ...
...
```

These are sketches of two DTDs. A DTD defines a set of tags, you could say typesetting instructions, and their hierarchy. The set of typesetting instructions is in fact a typesetting language. So in fact I've just given two typesetting languages. You could also define the syntax of a language like TEX in SGML. Mostly however, document type definitions are written with the logical structure of a class of documents in mind.

By the way: two parallel views of one piece of text—view 1: logical structure, view 2: visual structure—can be important or even essential in pre-existing text, something that is pointed out in the draft report of the Text Encoding Initiative [10], on which Michael Sperberg-McQueen will speak [9]. For example: inscriptions found on historical sites or texts in *real* manuscripts—you know: hand-written books.

At present however, publishers do not receive a great quantity of SGML-coded material—not yet! There are not many SGML editors available and the ones that are available are not or hardly ever used by the authors one finds in normal textbook or journal publishing. Furthermore, the word processors these authors use do not have an SGML export facility. So if a publisher wants to have material available in some form of SGML, it means converting it from whatever form the material is in when he receives it—at least for many years to come.

Encoding a piece of text with SGML means

- separating form from content, presentation from function
- adding structure to a text, enriching the text

In particular, the last activity is a time-consuming one, both for the author and the publisher, but it significantly increases the potential usefulness of the information. If a text is fully tagged, as it is called in SGML, if pieces of text are identified by their function, all sorts of information can be extracted, stored and re-used. For example: the article opening and the lists of literature references. If you use the text as part of a hypertext, links to figures, tables, references, footnotes and other parts of the text can be derived *automatically*.

But I would like to stress that SGML has nothing to do with getting a piece of text on paper or on screen. For that, you always need a separate program. So, 'SGML or TEX' is not a question at all, since you can't compare SGML and TEX. Valid questions to be asked are:

- do you combine SGML and TEX, SGML and Ventura, or SGML and you-name-it?
- *how* do you combine, let's say, SGML and TEX?

Suppose you use TEX as a back-end to a document-preparation system based upon SGML. What sort of problems do you encounter then? If you make a list of these problems and add ideas from various other TEX experts, you get a very long wish list indeed. What extensions do we need to add to TEX? *Are* we going to change TEX or are we going to build a completely new program?

3 Future of TEX

I'd like to spend a few minutes of my talk on this subject, since I'm not really happy with the current status of TEX. If you think the following is a bit provocative, well . . . , maybe it's intended that way.

To put it simply: I think the program should never have been frozen. Its author should either have continued developing TEX or handed over this work to a new implementor, or preferably a group of implementors. If this happens with professional—or, if you like, commercial—software, if you do not listen to the users of your program, or if you freeze a program, the software will be as good as obsolete after a few years.

I will not try to improve upon Frank Mittelbach's excellent paper 'E-TEX: Guidelines for Future TEX Extensions' [8], which he presented at last year's meeting in Texas. Rather, I will add a few of my own comments, or observations.

A big deficiency in TEX is the page-breaking algorithm and the tools TEX offers to program complicated page layouts, for example two-column or three-column with footnotes and floating bodies of 1 or more columns. If you use TEX as it now is as the back-end to an SGML-based system, page layout cannot be achieved fully automatically: manual work is still required. And even though TEX is intended to be used by a typist, not as a fully automatic back-end system, the more work the computer does without human intervention, the better. This makes the SGML–TEX combination far from ideal.

The same problem occurs if you use LATEX, which has a pretty complex output routine for scientific journals with a two-column layout, with lots of figures, tables and footnotes.

TEX users who have tried it know how difficult it is to let TEX typeset text—let's assume ordinary left-to-right text—in a language with lots of accented letters, ligatures and complicated hyphenation. Why are there no under-accents, multiple accents? Why is hyphenation of accented words or compound words with hypehns such a problem? I will use a few technical phrases from my own background, nuclear physics, as examples to show that the problem of hyphenating compound words, for example, is not just a problem of, say, the German or Dutch language.

Compound words are quite frequent in Dutch, for example:

```
schillenmodel-berekening
```

(shell-model calculation). Most TEX users would like to see TEX hyphenate this as 'schil-len-model-bere-kening', which TEX of course doesn't do.

But compound words of this type also occur in English:

```
formation of a compound nucleus
```

is hyphenated by TEX as 'for-ma-tion of a com-pound nu-cleus', whereas

```
compound-nucleus formation
```

is hyphenated by TEX as 'compound-nucleus for-mation', instead of 'com-pound-nu-cleus for-ma-tion'.

There should have been a switch for this in TEX, but there isn't! Why wasn't the functionality of TEX–XET and everything else I've mentioned added to TEX 3?

It is my opinion that TEX would have been a better program if its creator had agreed to re-think certain choices he had made years ago, especially when users argued their case by showing what sorts of problems TEX poses, as was done by several of them in articles in *TUGboat*. Barbara Beeton explained to me some time ago that the decisions regarding TEX's accent mechanism—`\accent` or ligature, single or multiple accents, only above or also below and to the side?—were Don Knuth's decisions and his only; they were not based on discussions with other experts, which I think is unfortunate. I sometimes think—and this is not intended as a bad joke!—that certain parts of TEX would have looked different if Knuth had been German or Greek, because English is such an easy language to typeset, relative speaking!

And while TEX is superior in mathematical typesetting, there is still a lot to criticize in that area as well. An example is the spacing between the eight basic types of math atoms, which is hardwired into the program as a sort of matrix, instead of being accesible via parameters. This results in a lot of handwork if a particular house style deviates from TEX's rules. Again, I would like to refer to Frank Mittelbach's article and the work on $\mathcal{AMS}$-TEX by Michael Spivak.

Another example: where's the missing lowercase Greek?

	upright form	slanted form
lowercase letter	?	π
uppercase letter	Π	$\mathit{\Pi}$

In other words: why was it arbitrarily decided that there was no need for upright Greek lowercase letters in the Computer Modern fonts?

A lot of work still needs to be done. Whoever is going to do it, I think that the successor to TEX 3—the matter of the name, TEX 4 or E-TEX 1 or God-knows-what, is unimportant, the important thing is that there should be *one* successor, not several incompatible systems based on or derived from TEX—should not be developed and maintained by

- one single person
- one or more persons all working in one field of work, for example mathematics or physics

  - otherwise the successor to the font set that is now more or less standard, Computer Modern plus $\mathcal{A}\!\mathcal{M}\!\mathcal{S}$-Fonts, will contain exotic symbols such as $\precnapprox$ and $\leftrightsquigarrow$, but not basic ones like the male and female symbols
- one or more persons all speaking the English language

During this conference there will be a panel 'The future of TEX'. An important subject, something the TUG board, TUG members and TEX users in general should think about a lot. As I said earlier: in principle, TEX has great potential for authors in *all* scientific disciplines and *all* languages, but only if the program is developed further.

4 Future of Publishing

The last topic I would like to talk about is the future of publishing. I don't think I am the right person to make prophecies concerning the future of publishing. Instead, I would like to present some ideas I have found in recent science fiction stories and novels.

One of the most striking ideas I've come across in the past couple of years is the idea of direct brain-computer coupling, as used by the Canadian author William Gibson, who is called the founder of the sub-genre 'cyberpunk', in his *Neuromancer* novels. With the direct brain-computer coupling, you can access any collection of data and it is as if you navigate with a virtual body through the space of data, which Gibson called 'cyberspace'. It is not such a weird idea at all, although an idea of the far future, and it is related to what people call 'virtual reality', a very popular phrase in some circles nowadays.

An idea that might become reality in the near future can be found in a book by the American science fiction writer, David Brin, in his latest novel 'Earth' [1]:

> If only it were a modern document, with a smart index and hyper links stretching all the way to the world data net. It was terribly frustrating having to flip back and forth between the pages and crude flat illustrations that never even moved. Nor were there animated arrows or zoom-ins. It completely lacked a tap for sound . . . in a normal text you'd only have to touch an unfamiliar word and the definition would pop up just below. Not here though. The paper simply lay there, inert and uncooperative.

To leave fiction and come back to the here-and-now: according to the Faxon Planning Report 1992 [3], Faxon Press'[1] poll of 52 periodical publishers, half of them commercial publishers, the other half non-profit organizations, a small majority of these publishers were quite worried about the future of publishing as we know it. Almost all of them still believe in the primacy of printed books and journals for decades to come. Is the vision David Brin presents something of the very far or of the very near future?

Just a few points to think about:

1. There are still librarians and scientists who see nothing whatsoever in electronic journals and books.
2. But the amount of information printed on paper increases exponentially.
3. And finding the right information becomes increasingly difficult.
4. Furthermore, increase of paper usage is also a serious environmental problem.

Well, you can't halt progress: electronic books are here already and their number will grow. In the transition period there is still another problem. An electronic book has to be available in paper form as well, since most readers still prefer a paper book.

Suppose you use TEX for the paper version, what do you use for the electronic version? How do you handle the two presentation styles? This is something I hope John Lavagnino will address in his talk on simultaneous electronic and paper publication of Thomas Middleton's complete works.

Is DSSSL[2] the answer to these problems, or FOSI[3]? What will the role of TEX be in non-paper publishing? I really don't know, but we should all think about it.

TEX is superior compared to desktop-publishing programs. It can handle mathematical formulas and complex tables, and this is a capability that is often lacking or poorly developed in desktop-publishing programs. Existing programs for the creation of electronic books also lack these capabilities: they can handle only text and graphics. If you want to include mathematical formulas or tables, the most sophisticated you can do is prepare bitmaps of these components—by means of scanning, or perhaps TEX?—and put these in the electronic document in the form of graphical objects.

5 Conclusion

This conference offers a great opportunity for discussions between TEX users and commercial professionals, since the programme contains a lot of talks about many different current applications. There are interesting panel discussions and hopefully there will be plenty of

[1] A large, completely automated subscription agent in the United States, involved in many activities.

[2] An ISO standard under development for the specification of document processing, such as formatting and data management [7]. The acronym stands for 'Document Style Semantics and Specification Language'.

[3] See the paper by Andrew Dobrowolski in these proceedings.

time for discussions during the breaks and in the evenings.

One of the goals of this conference is to try and bridge the gap—apparent or real—between the two poles of my first dichotomy: the author who is a TEX user, and publishers or other commercial professionals who want to accept TEX material. Looking at and thinking about present applications of TEX, as well as an historical perspective, can help to bridge this gap.

This conference is also a good opportunity to discuss the future of TEX, the future of publishing and the future of TEX-in-publishing. And I hope that it will be a success in all respects: that we will be able to find solutions to the problems I mentioned and those that will be described in the next four days—that we will be able to make gates in the fences and not just sit on the fences.

I'd like to thank the organization for inviting me to give this introductory talk. It was a pleasure to prepare and give this talk, and I feel honoured having been invited here.

References

[1] Brin, David. *Earth*. New York: Bantam Spectra, 1991. [Quoted with permission from the author.]

[2] Bryan, Martin. *SGML: An Author's Guide to the Standard Generalized Markup Language*. Workingham, UK: Addison Wesley, 1988.

[3] *The Faxon Planning Report 1992*. Boston: The Faxon Press, Boston, 1991.

[4] Herwijnen, Eric van. *Practical SGML*. Dordrecht: Kluwer Academic, 1990.

[5] Humphreys, Christmas. *Zen Buddhism*. London: Unwin, 1984.

[6] International Standard ISO 8879: Standard Generalized Markup Language (SGML). Geneva, 1986.

[7] International Standard ISO 10179: Document Style Semantics and Specification Language (DSSSL). Geneva, 1991.

[8] Mittelbach, Frank. "E-TEX: Guidelines for Future TEX Extensions." *TUGboat* 11(2), pages 337–345, 1990.

[9] Sperberg-McQueen, C.M. "Specifying Document Structure: Differences in LATEX and TEI Markup." See elsewhere in these proceedings.

[10] Sperberg-McQueen, C.M., and Lou Burnard, eds. *Guidelines for the Encoding and Interchange of Machine-Readable Texts*. Chicago and Oxford: Text Encoding Initiative, November 1990. [Available upon request from the authors.]