Nederlandstalige TEX Gebruikersgroep

MAPS: $M_{\text{inutes and }}AP_{\text{pendice}}S$ ..... #14 (95.1)

Verslag:

Bijlagen:

Nederlandstalige TEX Gebruikersgroep (NTG)

Voorzitter:	J.L. Braams E-mail: j.l.braams@research.ptt.nl	
Secretaris:	G.J.H. van Nes ECN, Unit Faciliteiten, Petten E-mail: vannes@ecn.nl	
Penningmeester:	W. Dol LEI-DLO, Den Haag E-mail: w.dol@lei.agro.nl	
Bestuursleden:	E.H.M. Frambach RUG, Econometrie, Groningen E-mail: e.h.m.frambach@eco.rug.nl	F. Goddijn E-mail: goddijn@fgbbs.iaf.nl
Postadres:	Nederlandstalige TEX Gebruikersgroep Postbus 394 1740 AJ Schagen	
Postgiro:	1306238 t.n.v. Penningmeester NTG Eindhoven	000-1662209-17 t.n.v. Ph. Vanoverbeke (NTG) Langenhoekstraat 28, B-8210 Veldegem, België
E-mail bestuur:	ntg@nic.surfnet.nl	

TEX is een, door professor Donald E. Knuth ontwikkelde, 'opmaaktaal' voor het letterzetten van documenten, een documentopmaaksysteem. Met TEX is het mogelijk om kwalitatief hoogstaand drukwerk te vervaardigen. Het is eveneens zeer geschikt voor formules in mathematische teksten.
Er is een aantal op TEX gebaseerde produkten, waarmee ook de logische struktuur van een document beschreven kan worden, met behoud van de letterzet-mogelijkheden van TEX. Voorbeelden zijn LATEX van Leslie Lamport en $\mathcal{AMS}$-TEX van Michael Spivak.

De **Nederlandstalige TEX Gebruikersgroep (NTG)** is een vereniging die tot doel heeft het bevorderen van de kennis en het gebruik van TEX.
De NTG tracht dat te bereiken door het uitwisselen van informatie, het organiseren van congressen en symposia m.b.t. TEX en 'TEX-produkten' en door het onderzoeken en vergelijken van TEX met soortgelijke/aanverwante produkten.

De NTG biedt haar leden onder meer het volgende:

- Tweemaal per jaar een NTG-bijeenkomst.
- Tweemaal per jaar het uitgebreide NTG tijdschrift: MAPS (Minutes and APpendiceS).
- Speciale MAPS uitgaven (o.a. TEX cursus en PR set).
- De *dubbele* 4allTEX CD-ROM met een volledige en direct te gebruiken TEX DOS/Windows/OS-2/Linux implementatie inclusief een uitgebreide verzameling van utilities. De CD-ROM bevat zeer veel documentatie, inclusief discussielijsten van vele jaren, *alle* MAPS uitgaven, en zeer veel tutorials.
- Korting op (buitenlandse) TEX congressen en cursussen en op het lidmaatschap van TUG.
- Jaarlijks een ledenlijst met informatie over welke software/hardware, in relatie met TEX, wordt gebruikt.
- De discussielijst (listserver) TEX-NL waarop vragen worden gesteld en ervaringen uitgewisseld.
- De fileserver TEX-NL waarop algemeen te gebruiken 'TEX-produkten' staan. De meeste van deze TEX-produkten zijn, tegen geringe vergoeding, ook op diskette verkrijgbaar. Daaronder valt ook de 4allTEX distributieset: een gebruikersvriendelijke en 'volledige' LATEX/TEX implementatie voor MS-DOS systemen.
- Het NTG FGBBS Bulletin Board met ruim 800 MByte aan TEX en aanverwante software.
- Aktiviteiten in werkgroepen. Enkele belangrijke werkgroepen zijn: 'Nederlandse TEX', 'PC's en TEX', 'educatie' (cursussen), en 'communicatie'.

Lid worden kan door overmaking aan de penningmeester van het verschuldigde contributiebedrag. Daarnaast dient een informatieformulier te worden ingevuld, dat via het secretariaat te verkrijgen is.

De contributie voor een persoonlijk lidmaatschap bedraagt *f* 90,–, de contributie voor een instituutslidmaatschap bedraagt *f* 245,–. Een instituutslidmaatschap geeft het recht om drie personen aan te wijzen die informatie welke aan de leden wordt verstuurd, ontvangen. Van die drie personen dient één persoon te worden aangewezen als rechtsgeldige vertegenwoordiger van het bedrijf/instituut, een ander als vervangend vertegenwoordiger. Indien meer leden per bedrijf/instituut lid willen worden, geldt als additioneel tarief *f* 65,– per persoon. Voor studenten geldt een tarief van *f* 60,– (geen stemrecht; bewijs van inschrijving vereist). Voor afwijkende regelingen dient contact met het bestuur opgenomen te worden. Een gecombineerd NTG/TUG lidmaatschap voor 1994 bedraagt *f* 167,50 per jaar (i.p.v. *f* 90,- + $ 55).
Belgische leden kunnen de lidmaatschapskosten van BF 1660 (individueel), BF 4520 (instituutslidmaatschap) of BF 3090 (NTG/TUG lidmaatschap) overmaken op de NTG Belgische postgiro te Veldegem (zie hierboven).

Nederlandstalige TEX Gebruikersgroep

Aanwezig : A. Al-Dhahir (UT); L. Bastiaenssen (UIA); A.W.W.M. Biegstraaten (TUD); P. Bloemen; T. Bloo (De Jonge Onderzoekers) H. Blühme (UIA); J. Braams; F. Claeys (Univ. Gent); L. de Coninck (de Kraal); N. Cox (KUN); W. Dol; E. van Eynde (K.U. Leuven); E.H.M. Frambach (RUG); F. Goddijn (FGBBS); M. Goossens (CERN); W.J. van de Guchte; H. de Haan; J. Hagen (Pragma); J. van Hamme (Kon. Militaire School); S. van Harreveld; R. van der Heijden (Hogeschool Utrecht); A. Heijs (Staring Centrum); D. van Heule (Kon. Mil. School); E. de Hoover (Editex); H. Janssen (Kon. Mil. School); R. de Jeu (Wolters Kluwer); C.R. Kloos (Scan Laser B.V.); J. Krugers (T.M.C.); A. de Leeuw van Weenen (RUL); F.D. Mesman (Elsevier Science Publishers); G.J.H. van Nes (ECN); G. Oomen (Wolters Kluwer); P. van Oostrum (Universiteit Utrecht); S.A.M. Pepping (Elsevier Science Publishers); F. Ploeschoert (Editex); J. Pijnenburg (KUB); J. Renkema (Theol. Univ.); A. van Ryckeghem; L.B. Schelhaas; A.C.A. Serier; W. Smit; A. Soos (UT); R.C.W. Strijbos (UT); B. Suykerbuyk (B+B Uitgeverij); P. Tutelaers (TUE); E. Ulijn (TUD); P. Vanoverbeke; B. Verhegghe (Universiteit Gent); G. Vlak; B. Wage (Elsevier Science Publishers); S. van Wakeren; J. van Weelden; J.E. van Weerden (Universiteit Utrecht); B. IJff (Scan Laser B.V.),

Notulist : Frans Goddijn

1 Opening

Bij aanvang van deze *eerste NTG-bijeenkomst in België* geeft penningmeester Wietse Dol informatie over de geldende wisselkoers voor het afrekenen van de diverse lunchmogelijkheden in de Universitaire Instelling Antwerpen. Benoit Suykerbuyk, onze gastheer, vertelt welke deze (uitgebreide!) mogelijkheden zijn.

2 Verslag van de NTG-bijeenkomst van 9 juni 1993

Jules van Weerden meldt, als opmerking bij bladzij 3, dat er ook een HTML BROWSER voor DOS bestaat (DOSLYNX). Voorts stelt Jules voor om bij NLNET het domein NTG.NL te claimen. Johannes Braams vindt dit een goed idee en zal hiervoor gezwind actie ondernemen.
De notulen worden aldus vastgesteld.

3 Ingekomen stukken en mededelingen

Gerard van Nes meldt dat er vele bestellingen voor de NTG CD-ROM zijn ontvangen en verwerkt.
Van Henk de Haan is een voorstel ontvangen om te komen tot een WWW-site voor het NTG.
De diverse TEX zusterverenigingen hebben het NTG verblijd met diverse publicaties, waarbij in het bijzonder de kleurenuitgave van GUTenberg valt te bewonderen op de zeer gevulde leestafel.
Johannes Braams heeft als NTG-deelnemer reeds de beschikking over de *proceedings* van de TUG'94 bijeenkomst te Santa Barbara. Johannes wijst er nog op dat Donald Knuth begin 1995 weer een blik zal slaan op eventueel binnengekomen bug-reports. Wie van plan is er één in te sturen: aarzel niet langer.

4 Begroting 1995

Vanwege een zeer late overdracht van het penningmeesterschap kon de concept NTG-begroting 1995 niet meer in de MAPS worden opgenomen. Een afdruk op papier wordt aan de aanwezigen ter beschikking gesteld.
Wietse Dol, de nieuwe penningmeester, licht toe waarom er een geringe verhoging van de contributie voor 1995 aan zit te komen.
Piet van Oostrum benadrukt dat er, zoals ook in het verleden, enige coulantie moet zijn voor leden die problemen hebben met het voldoen van hun contributie, zoals studenten.
Toin Bloo vraagt zich af hoe het zit met de goede oogst van het 4allTEX CD-ROM project. Is daardoor een verhoging van de contributie niet overbodig geworden? Wietse Dol geeft aan dat het succes van dit project geldt als *incidentele* meevaller, hetgeen in principe los staat van onder meer de kosten voor het maken van de MAPS en andere zuivere NTG aangelegenheden.
Alaaddin Al-Dhahir merkt op dat een geplande kleuren-

Editors van deze MAPS zijn: Wietse Dol, Frans Goddijn, Gerard van Nes, Philippe Vanoverbeke en Jos Winnink.
Oplage MAPS: 325.
Het verslag van de NTG bijeenkomst op 17 november 1994 is (in concept) begin maart 1995 via de post reeds gestuurd naar alle NTG leden.

uitgave van de MAPS niet per se nodig is, waarop Gerard van Nes toelicht dat dit voor één keer wél nodig is, aangezien de komende MAPS in principe een *special* over kleur gaat worden.
Henk de Haan verzoekt of er in de toekomst een uitsplitsing kan worden gemaakt van de kosten voor bestuur en bijeenkomsten.
Phons Bloemen vraagt wat er valt onder de post *afschrijvingen*, waarop Wietse Dol aangeeft dat het hier de HP LJ 4MP printer van het NTG betreft.

Er volgt een stemming over de begroting, en deze wordt aangenomen met 6 tegenstemmen, 4 onthoudingen (de rest is vóór).

5 Wat verder ter tafel komt

Gerard van Nes verklaart de kleurverschillen in de badges: rood is bestuur, blauw zijn sprekers, wit zijn de leden en groen de gastheren (Suykerbuyk, De Coninck, Vanoverbeke). Zo weet men elkaar te vinden.
Gemeld wordt dat de NTG op dit moment 262 leden heeft waaronder 35 instituutsleden en 10 studentleden. Het aantal TUG/NTG-leden bedraagt 81.

Addison-Wesley is aanwezig met een uitgebreide boekentafel met onder meer TEX/LATEX en PostScript boeken (de tweede druk van de fameuze Companion is vrijwel uitverkocht!). Addison-Wesley is ook de eerste donateur van NTG getuige de advertentie in de MAPS.

Gerard van Nes vertelt over de MAPS-produktie. Deze vindt nu plaats in Groningen, bij dezelfde drukker die de manual bij de NTG 4allTEX CD-ROM heeft gemaakt. Dit heeft opnieuw een iets betere kwaliteit opgeleverd en de volgende MAPS zal volgens plan weer iets scherper gedrukt zijn, geheel met 4allTEX opgemaakt en op speciaal papier.
Jules van Weerden vraagt of hierbij kan worden gelet op het gebruik van chloorvrij papier. Erik Frambach zegt toe hier aandacht aan te zullen besteden.

Gerard van Nes meldt dat er weer een NTG MAPS Award is toegekend. Ook ditmaal waren er weer velen die zich bijzonder verdienstelijk hebben gemaakt, waardoor een goede MAPS kon worden uitgebracht. De keuze is uiteindelijk gevallen op de bijdrage van de *heer Hagen* van *Pragma*, vanwege onder meer het bijzondere onderwerp, en het vele daarachter liggende TEX researchwerk. Daarnaast was het de eerste bijdrage die deze, voorheen 'onbekende' auteur, aan de MAPS redactie aanbood.

Voor wie dit nog niet wist: de NTG MAPS Award bestaat uit een reusachtige puntzak gevuld met `stretchable boxes`.

Johannes Braams kondigt aan dat de nieuwe versie van LATEX2ε kan worden tegemoetgezien aan het eind van 1994. Voorts meldt hij dat het NTS-team in fasen verder werkt aan de beoogde opvolger van TEX. Eerst zal op afzienbare termijn eTEX worden uitgebracht, *enhanced* TEX, nog geheel compatibel met TEX. Door middel van *switches* zullen extra faciliteiten kunnen worden aangeboden. De daarop volgende versie zal geheel een *redesign* vormen, als prototype in LISP uit te voeren.
Bovendien is Yannis Haralambous druk met het reeds in kleine kring bekende *Omega-project* dat op zijn eigen manier een opvolger van TEX zou kunnen worden.
Piet Tutelaers vreest door deze verscheidenheid aan 'opvolgers' een versnippering van ontwikkelarbeid. Hierop ontspint zich een deels technische, deels principiële discussie over de wenselijkheid van toekomstige 'features' in de opvolgers van TEX, de problematiek van Postscript fonts, DC fonts, VF fonts en 16-bits UNICODE. Geopperd wordt dat de kwaliteit van eTEX later in NTS zou kunnen worden ingebouwd. In plaats van een versnippering is er eerder sprake van fasering: wat eTEX kan bieden (16 bits UNICODE, vele talen o.a. Indisch en Koreaans) komt over enige tijd beschikbaar, terwijl het veel ambitieuzere NTS pas op lange termijn gestalte zal krijgen. In dit verband vraagt Piet Tutelaers wat de visie is van Leslie Lamport, zoals deze is weergegeven tijdens Lamports lezing op de TUG-bijeenkomst in Santa Barbara. Michel Goossens licht toe dat de lezing van Leslie Lamport bijzonder interessant was en uitermate provocatief, maar dat hetgeen hij te berde bracht in het geheel niet implementeerbaar is.

Wietse Dol doet verslag van het bezoek aan EuroTEX. Het was met name leuk om de 'grote jongens' uit het LATEX-circuit eens in levenden lijve te ontmoeten (afgezien van onze eigen Johannes Braams, die we al kennen). Daarbij was opvallend dat de Poolse TEX-gebruikers buitengewoon inventieve mensen zijn, niet alleen op TEX-gebied maar ook in de andere aspecten van het leven. Academici in de voormalige oostblok-landen moeten alle zeilen bijzetten om economisch te overleven in deze voor hun landen zo zware tijden.
4allTEX in de versie op diskette is er zeer populair, wat aansluit bij de noodzaak om méér te doen met minder middelen. Dit brengt voor het 4allTEX team de verplichting mee om ook tijd te blijven besteden aan een *update* van deze 4allTEX collectie op diskette. Een van de voornemens op dit gebied is het uitbrengen van een nieuwe versie op .ZIP-formaat in plaats van het tot nu toe overheersende .ARJ-formaat.
Gerard van Nes roemt het EuroTEX-verslag van Michel Goossens: dit was *binnen 24 uur* na afloop van de conferentie reeds beschikbaar en kon nog net in de MAPS worden opgenomen.

Michel Goossens sluit zich aan bij hetgeen Wietse Dol vertelde over de penibele situatie in de oostelijke TEX-gewesten; in Rusland is voor veel TEX-users niet eens *papier* voorhanden om teksten op af te drukken. De CYRTUG uitgeverij heeft 80% van de mensen moeten ontslaan door dit gebrek aan eerste grondstof. Het bizarre is dat boeken over honden en katten nog wel kunnen worden gedrukt voor de export, terwijl de verschijning van eigen wetenschappelijke werken tot minder dan 5% is teruggevallen. Daarbij is het distributienetwerk van voorheen verdwenen. Gevolg van dit alles, genomen bij de inflatie van 20%, is dat de economische levensduur van een boek voor de eigen markt ultra-kort is geworden. Raakt het boek niet direct

uitverkocht, dan is het verlies al bijna niet meer te overzien. Alternatief zou het data-netwerk kunnen zijn, ware het niet dat de daarvoor benodigde communicatiemiddelen (telefoonlijnen!) alles te wensen overlaten. Het zou in dit licht een goed idee zijn om met alle LUG's samen een CD-ROM te produceren, waarop ook alle daarvoor in aanmerking komende artikelen konden worden geplaatst.

6 Communicatie

6.1 Communicatie: FGBBS

Henk de Haan, cosysop van FGBBS, geeft middels de PC van de NTG-secretaris toelichting op zijn plannen voor een NTG WWW-site. Op het grote scherm worden voorbeelden getoond van wat er nu reeds aan is geproduceerd.
Frans Goddijn vertelt dat er net een nieuwe grotere hard disk is bijgeplaatst op de computer van het FGBBS.

6.2 Communicatie: TEX-NL, CTAN

Jules van Weerden oppert zijn idee voor een zogeheten 'owner' TEX.NL. Ook zal Jules nog eens zijn licht laten schijnen op de regelmatig terugkerende gedachte om de TEX-NL lijst ook als *news* te brengen op `usenet`.

Johannes Braams geeft een korte uiteenzetting over de tarieven voor het TUG lidmaatschap. Lidmaatschap plus abonnement op TTN kost 40 dollar, de TUGboat kost 15 dollar, lidmaatschap geeft 20% korting op aankopen via TUG en een institutioneel lid betaalt 350 dollar voor 7 personen.

6.3 EuroTEX in Nederland!(?)

Na het 'besluit' hiertoe, in Polen genomen, is het bij de organisatoren in Nederland vrij rustig gebleven. Sterker nog, er zijn nog geen organisatoren. Erik Frambach zal op TEX-NL een oproep plaatsen om te peilen wie mee wil werken aan de totstandkoming van deze conferentie in Nederland. Ook zullen mensen persoonlijk worden benaderd. Piet van Oostrum is voorshands al bereid mee te denken. We zouden uit kunnen gaan van een opkomst van 75 NTG-leden, evenveel gasten uit het buitenland, en ten hoogste 250 deelnemers. De eerdere EuroTEX bijeenkomst in Polen telde 60 deelnemers; in Praag 200 deelnemers. De bijeenkomst zou kunnen plaatsvinden in het najaar van 1995 (de eerstkomende TUG-conferentie is in juli 1995).

7 Rondvraag en sluiting

Kees Serier stelt voor om de MAPS viermaal per jaar te gaan maken. Gerard van Nes wijst erop dat dit veel meer werk met zich mee zou brengen. Vrijwilligers zijn altijd welkom. Niet alleen voor de MAPS-werkzaamheden.

De volgende vergadering zal zijn in Twente! Johannes sluit met een ferme klap deze vergadering.

8 Voordrachten:

Het hierna volgende is, als tegemoetkoming aan de Engelstalige lezers van de MAPS, in het Engels gesteld!

A pleasant series of readings ensued, opened by Wietse Dol and Erik Frambach demonstrating the excellent state of affairs at the 4allTEX work bench development.

Then, from two Dutch multinational scientific publishers, Rob de Jeu (Kluwer) and Simon Pepping (Elsevier) each told about how LATEX has changed their publishing methods. Their articles can be found in the MAPS. The general and fascinating message was that, a few years ago, some authors of highly specialized scientific articles began sending in their texts in prints that were typeset beautifully. This caused surprise and curiosity — how did these scientists pull that off? Tactful probing learned that TEX had been at work here. Gradually, after setting up a trial LATEX system, some authors who were obviously using Computer Modern were asked to send in their source files and as the production people got more and more comfortable with LATEX, other authors were also introduced to LATEX. Thus, gradually, the typesetting work at the publisher's desk was partially moved towards the author's desk.

Now, every year more than a hundred scientific magazines and thousands of book pages are published using LATEX by these two publishers world wide. The publishers, who were originally introduced to LATEX by the authors, have gradually regained the initiative by composing specific style files which remain in strict internal use for the final stage of publication, with simpler derivative style files for the authors who can use these as an envelope for their articles. To the publishers, LATEX has two outstanding secondary features which both readers mentioned smilingly: first, it is free, and second, excellent support for the program is *also* free (and available at lightning speed through the internet)!

But recently, with the availability of high quality printing devices and with more and more scientists arriving at true guru level in TEX, the publishers are beginning to get mail from scientists who offer their articles for publications *in a typography and layout identical to the magazine's standards*! This way, it's possible for some people to present their work *as if* it had already been published by a house of high standards. . .

Theoretically, therefore, authors could organize their own referee system and publish independantly from the established publishing companies without any loss in scientific or typographic quality.

The publishers still hold the prestige factor, as their referee system guarantees a quality of selection that can't be easily duplicated by others. Furthermore, research and development is done to convert the gigantic and ever growing file library of articles from LATEX into other systems, like HTML. Such a new article base could serve as input for publishing books on demand, for availability in World Wide Web sites and more.

After tea, Prof. H. Blühme lectured us, with intriguing slides, on his work at creating an etymological dictionary with TEX. The audience listened in fascination to Blühme's account of his work, which gave us insights in the life of

words, passing through borders between countries and creating a family of related words elsewhere.

Michel Goossens told us of the history that was written while he and fellow authors Samarin and Mittelbach collaborated at the writing of the now famous **LaTeX Companion**. When Michel is talking, you not only learn things about the general line of the subject, but you are amused, intrigued and educated by his many asides, his gentle erudition and inspiration. I don't think there is an essential group or development in TeX that he is not directly or indirectly involved with and any activity that he supports, benefits greatly.

Andrea de Leeuw van Weenen concluded the day with an account of the long and often bizarre adventure which led to her Icelandic book, typeset in TeX. Over a period of many years, every unbelievable production error was made by the linguistic authorities involved, time and time again wiping out all work done by Andrea. More than once an over-extensive editing job was rendered useless because the 'advanced' printing machine for which it was originally designed, had in the meantime become obsolete. This was only resolved when Andrea decided to do it all by herself, using TeX and Metafont to create the ancient Icelandic alphabet.

Dinner was served in the Amadeus Restaurant, where the specialty of the house was 'unlimited roasted ribs'. The direct vicinity of this famous dining place had a seemly unlimited amount of nude female hams on display, which explained why not one Antwerp citizen admitted to knowing the way to this excellent restaurant (St. Paulusplaats).

On the next day, Michel Goossens treated a group of about 16 participants to a course in LaTeX2ε. Some would state that a thorough reading of the latest edition of the Companion would suffice to the same effect, but those people missed out on some of the best of Michel's observations. The *petite histoire* behind the genesis of the `\emph` command for instance. It was also interesting to learn which technical guru details were (according to Michel) too complicated for even him to grasp, like the contents of certain font definition files. These come in so many different versions, and some 'new' versions are actually returns to versions that had been 'new' quite some time ago, that is is beyond the grasp of average mortals how `ptmbo0` and `ptmbiq` arrive to make life difficult. Karl Berry is one of the leading men in this field, but who can decide why a new version of the font naming scheme is engendered: because the former one was too difficult, or because it wasn't yet complicated enough? The difference in checksum calculation between `fontinst` and `afm2tfm` is another of these miracles which surprise us ceaselessly. And the classic case of the `rotating` package which began rotating in the other direction... It is clear that humor has remained in TeX's history since the TeXbook.

Besides these stories and looks behind screens, the well structured day gave everyone who attended a clear view on the benefits of converting to LaTeX2ε and a basic understanding of the implications on the necessary changes in their home and office LaTeX setup.

9 Volgende bijeenkomsten

De voorlopige planning van de volgende bijeenkomsten ziet er als volgt uit:

- **voorjaar 1995**
  Universiteit Twente op 24 mei 1995
  Onderwerp 'Hypertext & TeX'
- **najaar 1995**
  Papendal op 5 september 1995
- **EuroTeX'95**
  Papendal op 4–8 september 1995

1 TEX kalender 1995

24–28	jul '95	TUG '95	St. Petersburg, Florida, USA
4–8	sep '95	EuroTEX '95	**Papendal, Arnhem**
5	sep '95	NTG (16^{e})	**Papendal, Arnhem**

2 Glossary

Gebruikersgroepen

TUG	:	TEX Users Group
LUG	:	Local Users Group
CSTUG	:	LUG Tsjecho Slowakije
CyrTUG	:	LUG USSR (het Cyrillisch taalgebied)
DANTE	:	LUG Duitsland (het Duits taalgebied)
GUTenberg	:	LUG Frankrijk (het Frans taalgebied)
HunTUG	:	LUG Hongarije
ITALIC	:	LUG Ierland
JTUG	:	LUG Japan
Nordic	:	LUG Scandinavië, Denemarken, en IJsland
NTG	:	LUG Nederland en België
SibTUG	:	LUG Siberië
UKTUG	:	LUG Engeland
YUNUS	:	LUG Turkije (feitelijk een discussielijst)
GUST	:	LUG Polen

Bulletins/journals

Baskerville	:	UKTUG
Cahiers GUTenberg	:	GUTenberg
Zpravodaj	:	CSTUG
TeXnische Komödie	:	DANTE
TeXline	:	Malcolm Clark; UK
GUST bulletin	:	GUST
TTN	:	TEX and TUG News; TUG
TUGboat	:	TUG
MAPS	:	Minutes and APpendiceS; NTG

Diversen

CTAN	:	Comprehensive TEX Archive Network; sites waar men 'anonymous ftp' kan gebruiken om TEX/LATEX-achtig materiaal te verkrijgen. CTAN is de 'home' voor de officiële versie van LATEX etc. CTAN sites zijn: `ftp.dante.de`, `ftp.tex.ac.uk` en `pip.shsu.edu`
FGBBS	:	NTG's Bulletin Board
AllTEX	:	TEX, LATEX, etcTEX
ltxiii	:	LATEX 3.0
4TEX	:	Het volledige TEX *runtime systeem* voor MS-DOS PC systeem, gebaseerd op emTEX en 4DOS.
4allTEX	:	De 4TEX applicatie *plus* alle mogelijke gerelateerde files en utilities, gedistribueerd op de diskette set en CD-ROM.
AMS	:	American Mathematical Society
BoD	:	Board of Directors
SGML	:	Standard Generalized Markup Language

3 NTG's TEX Bulletin Board Systeem

Op het TEX Bulletin Board van de Nederlandstalige TEX Gebruikersgroep (FGBBS) is een zo volledig en actueel mogelijke TEX, emTEX, LATEX, `TEX-NL` en MusicTEX collectie beschikbaar voor alle bezitters van een modem. Het BBS is kosteloos toegankelijk voor iedereen en er zijn geen beperkingen aan de hoeveelheid bestanden die kunnen worden opgevraagd. Het systeem is aangesloten op een High Speed modem, vergeleken met de transmissiesnelheid die een directe Internet link biedt misschien niet geweldig, maar veel beter kan het niet over de gewone huistuin- en keukenPTTlijn. De beheerders zijn Frans Goddijn en Henk de Haan. FGBBS is te bellen op 085-217041.

4 NTG's winkel

Via de NTG is beschikbaar:

- **4allTEX dubbel CD-ROM:**
  Ruim 1.1 Gbyte (ruim 50.000 files) aan onder meer 4TEX utilities, fonts, TEX/LATEX/METAFONT/etc documentatie, de volledige MAPS 1 t/m 14, discussielijsten `TEX-NL`, `TEX-HAX`, `UKTEX` van de afgelopen 6 jaren, etcetc.
  Kosten inclusief het uitgebreide 4TEX manual: *f* 60,–.
- **Syllabus Advanced TEX course:**
  Insights and Hindsights, David Salomon (*revised*; ruim 500 pagina's).
  MAPS'92 speciale uitgave.
  Kosten *f* 60,– voor leden en *f* 70,– voor niet-leden (extra verzendkosten: *f* 15,–).
- **NTG PR set**
  Enkele mogelijkheden van (LA)TEX(ruim 25 pagina's).
  MAPS'93 speciale uitgave.
  1 exemplaar gratis voor leden; extra exemplaren: *f* 7,50; niet-leden: *f* 15,– (inclusief verzendkosten).
- **100 Frequently Asked Questions**
  'Alles' wat u minstens moet weten over TEX, LATEX, METAFONT, PostScript, Special typesetting, Format conversions, DVI Drivers, Previewers, recente TEX ontwikkelingen, en nog veel meer. . . (24 pagina's).
  MAPS'94 speciale uitgave.
  1 exemplaar gratis voor leden; extra exemplaren: *f* 7,50; niet-leden: *f* 15,– (inclusief verzendkosten).

Bestellingen kunnen gedaan worden door overmaking van het verschuldigd bedrag (plus verzendkosten) op de postgiro van NTG (1306238) t.n.v. penningmeester NTG, Eindhoven, met vermelding van hetgeen gewenst is.

```
<A HREF="http://www.ucc.ie/info/TeX/tug/tug.html">                          The TeX Users Group  </A>
<A HREF="http://www.cl.cam.ac.uk:80/UKTUG/">                                The UK TeX Users Group  </A>
<A HREF="http://www.dante.de/">                                             DANTE e.V. - WWW-Intro  </A>
<A HREF="http://www.ens.fr:80/gut/">                                        L'association GUTenberg  </A>
<A HREF="http://www.cogs.susx.ac.uk/cgi-bin/texfaq2html?introduction=yes"> TeX Frequently Asked Questions  </A>
<A HREF="http://www.cogs.susx.ac.uk/users/alanje/latex/">                   LaTeX: A document preparation system  </A>
<A HREF="http://www.tex.ac.uk/TeXdoc/TeXdocs.html">                         TeX-related documentation  </A>
<A HREF="http://xxx.lanl.gov/hypertex/">                                    HyperTeX  </A>
<A HREF="http://www.YandY.com/">                                            Y&Y Inc Home Page  </A>
<A HREF="http://www.ens.fr:80/omega/">                                      Omega Home  </A>
<A HREF="http://www.ucc.ie/cgi-bin/ctan">                                   CTAN Search  </A>
<A HREF="http://www.shsu.edu/cgi-bin/ctan-index">                           ftp.SHSU.edu CTAN Index  </A>
<A HREF="http://jasper.ora.com/ctan.html">                                  CTAN-Web Home Page  </A>
<A HREF="http://www.cs.ruu.nl/people/otfried/html/hyperlatex.html">         The Hyperlatex package  </A>
<A HREF="http://www.tex.ac.uk/tex-archive/">                                Index of /tex-archive/  </A>
```

Figuur 1: *WWW TEX addressen*

5 Nieuw: Nederlandstalige LATEX handleiding

Net verschenen van de hand van LATEX guru Piet van Oostrum (`piet@cs.ruu.nl`): een uitgebreide LATEXhandleiding (92pp). Te vinden op `ftp.cs.ruu.nl:/pub/TEX/DOC/ltxhandl.zip`. Voorlopig als PostScript file, doch ook de source zal beschikbaar worden gesteld!

6 TEX op WWW

In figuur 1 vindt U een aantal URL adressen waar TEX achtige zaken te vinden zijn[1].

Bekijk ook eens de WWW pagina's van Blue Sky Research (`www.bluesky.com`) waarin opgenomen is een pagina met *'Interesting TEX-related URLs'*. Zeer de moeite waard!

7 NTG op WWW

Er wordt druk gewerkt om een aparte server voor de NTG op te richten voor NTG zaken. Het officiële adres zal worden: `http://www.ntg.nl`. Jules van Weerden is momenteel bezig een en ander bij zijn faculteit te Utrecht beschikbaar te maken.

8 NTG/TUG lidmaatschap

Het blijkt soms dat nieuwe NTG/TUG leden na ongeveer een half jaar nog geen TUGboat of TTN van TUG hebben ontvangen. Ondanks dat men een TUG lidmaatschap via NTG aanvraagt, blijkt in bijna alle gevallen de administratie- en verzendproblemen bij TUG zelf te liggen. Mocht na enige maanden tijd geen post van TUG ontvangen worden, dan worden de betreffende NTG/TUG leden dringend verzocht om kontakt op te nemen met Johannes Braams (`j.l.braams@research.ptt.nl`) of met het secretariaat van de NTG.

9 De NTG najaarsbijeenkomst

De NTG najaarsbijeenkomst 1995 zal op dinsdag 4 september tijdens de EuroTEX'95 bijeenkomst op Papendal te Arnhem plaatsvinden. Het formele gedeelte van de vergadering zal ongeveer 45 minuten duren. De rest van de dag worden de NTG leden in de gelegenheid gesteld de conferentie zonder kosten bij te wonen. De tarieven voor de conferentie voor de andere dagen zullen nog t.z.t. bekend worden gemaakt.

10 MAPS 95.2

Sluitingsdatum voor het inleveren van artikelen, bijlagen, en/of mededelingen voor de volgende MAPS uitgaven is: *15 juli '95 (MAPS 95.2; #15) en 1 april '96 (MAPS 96.1; #16).*

Daar de 1995 najaarsbijeenkomst samenvalt met de EuroTEX'95 te Arnhem, zal MAPS 95.2 de proceedings van de conferentie bevatten. Speciale Nederlandstalige bijdragen voor de MAPS 95.2 zullen tot een minimum beperkt blijven.

Aanleveren kopij voor de komende MAPS:

- *Bij voorkeur* in LATEX gebruikmakend van de:
  `maps.sty`
  Deze stijlfile is via de redactie te verkrijgen en beschikbaar op de `TEX-NL` fileserver, archive.cs.ruu.nl (ftp-site) en FGBBS (085-217041).
  Daarnaast kunnen bijdragen ingestuurd worden gemaakt met `ltugboat.sty` of `article.sty/report.sty`.
- Verder zijn bijdragen vanzelfsprekend ook welkom in *plain*-TEX of ongeformatteerd.
- Plaatjes bij voorkeur als (Encapsulated) PostScript file.

Daar MAPS bijdragen in plain *TEX altijd worden omgezet naar LATEX, verdient vanzelfsprekend aanbieding van materiaal in LATEX de voorkeur!*

Eventuele nadere richtlijnen voor auteurs zijn op te vragen bij de redactie.

Bijdrage kunnen gestuurd worden naar:
`vannes@ecn.nl`

Niet Internet-gebruikers kunnen hun bijdrage ook via modem/PTT lijn direct naar de redactie sturen. Gaarne hiervoor eerst kontakt opnemen met Gerard van Nes.

[1] Met dank aan Sebastian Rahtz voor het verstrekken van deze informatie.

Van de Voorzitter

Johannes Braams

In mijn vorige 'column' schreef ik over de relaties tussen TUG en de lokale TEX gebruikersgroepen: 'Ik verwacht dat we in het komende jaar een aantal belangrijke veranderingen zullen meemaken." Welnu in de achter ons liggende periode is op dit terrein het één en ander gebeurd.

De vertegenwoordiger van de Franstalige TEX Gebruikersgroep (GUTenberg) heeft zich na overleg met zijn bestuur terug getrokken uit het bestuur van TUG.

Naar aanleiding hiervan is overleg op gang gekomen tussen onze vereniging, de Britse vereniging UKTUG en de Skandinavische TEX gebruikersgroep over onze posities. In dat overleg heeft het bestuur van NTG voorgesteld dat de vertegenwoordigers van deze drie gebruikersgroepen zich *gezamenlijk* zouden terugtrekken uit het bestuur van TUG. Daarbij hebben we duidelijk aangegeven dat de motivatie hiervoor is dat op deze manier de onduidelijke rol van de 'special directors' (dat waren de voorzitters van DANTE, GUTenberg, NTG, UKTUG en Nordic TUG) in het bestuur van TUG hiermee ten einde komt en dat daarmee *alle* lokale TEX Gebruikersgroepen dezelfde uitgangspositie hebben bij het vinden van een (nieuwe) relatie met TUG. Ook hebben wij aangegeven dat wij van mening zijn dat de duale rol van TUG (zowel een wereldwijde gebruikersorganisatie als een lokale gebruikersgroep voor onder andere de VS) op termijn zou moeten leiden tot een splitsing van TUG in een wereldwijde organisatie en één of meer LUGs.

Dit voorstel van de NTG is overgenomen door onze zusterorganisaties. Hieronder is de integrale tekst van het standpunt dat wij hebben ingenomen, afgedrukt.

```
From Johannes Braams:  President, NTG
Dag Langmyhr:  Leader, Nordic TUG
Chris Rowley:  Chair, UKTUG
```

Statement

```
In the light of the recent withdrawal from
the board of TUG of the special directors
from both DANTE and GUTenberg, we have dis-
cussed our position between ourselves and
with our respective boards/"|committees.

We have decided that the future of the re-
lationships between all LUGs will be more
constructively discussed if all LUGs have
equal status; therefore it is best to end
our anomalous position as special directors
of TUG and to withdraw from being ex offi-
cio members of the board of TUG.

Thus we see this withdrawal as a positive
action to clear the ground for building a
new relationship between all the LUGs since
it will create a situation in which *all*
local user groups are in the same posi-
tion.  We think that it should become pos-
sible to build a truly international user
group structure from there.
```

The way forward

```
It is our opinion, supported by our boards/-
committees, that the future of TUG as an
international organisation serving the TeX
community would be greatly improved if the
two roles of TUG, as a world-wide organi-
sation and as a LUG for North America (and,
possibly, other countries), were seen to be
clearly separated.

We must emphasize that this is no reflec-
tion on the work of the TUG office, nor on
that of any individual TUG board members,
all of whom have an international outlook
and provide a lot of support to the world-
wide TeX community.

The current problems are in these ar-
eas:
  • the perception of many LUG members that
    TUG is a North American organisation;
  • both the legal status of TUG and its
    constitution appear to make it unable to
    become a more clearly international or-
    ganisation;
  • it is not clear that TUG itself, as cur-
    rently constituted, can become an organ-
    isation that links all LUGs.

We realise that this separation may not be
feasible to achieve in the very near fu-
ture, but it should be something that TUG,
working with all the LUGs, should aim to
achieve in the long run.
It is our belief that some very different
international structure is the only sound
basis on which healthy relationships be-
tween the various TeX user organisations
can continue to exist.
```

Ik hoop dat de NTG op deze wijze een constructieve bijdrage heeft kunnen leveren aan het oplossen van de impasse waarin TUG verkeert ten aanzien van haar houding tegenover lokale gebruikersgroepen.

Van uw MAPS Editor

Gerard van Nes

Het ponskaarten mysterie

Oftewel: *Holland op zijn smalst?*

Op maandag namiddag 31 october 1994, stapte Uw NTG-MAPS editor het postkantoor van Schagen binnen, gebukt onder het gewicht van 150 kg aan diverse soorten postpakketjes: MAPS #14.

NTG: Goede middag meneer, kunt U mij helpen, ik heb iets te versturen.

PTT: Dat kan, laat maar eens zien wat U heeft.

NTG: Ik heb onder meer 130 pakketjes met de MAPS voor onze NTG leden en die zou ik graag als *partijpost* willen versturen. Moet van onze zeer zuinige penningmeester, scheelt hem een kleine honderd gulden aan porto en mij scheelt het weer heel wat gelik. Ik krijg tegenwoordig al zoveel rommel binnen en mijn tong is ook niet van steen. Uw stempelman heeft het dan ook weer eens wat rustiger.

PTT: Dat kan helaas niet meneer. Voor *partijpost* is sinds onze laatste tarievenverhoging nu een minimum aantal van 250 stuks in plaats van 100 stuks nodig. U zal dus voor al Uw 130 pakjes postzegels bij het loket moeten kopen, deze dan vervolgens met Uw tong moeten bevochtigen en op Uw pakje plakken. Wij zullen daarna graag al Uw 130 pakjes bestempelen. Ook de PTT heeft regels. En regels zijn nu eenmaal regels.

NTG: Ja mijnheer de PTT, de tarievenverhoging is mij en onze penningmeester bekend. Ik wil daarom ook graag al deze pakjes als *belstukken* versturen. Ik lees in Uw tarievenboekje dat er dan voor partijpost slechts een *minimum aantal van* 25 *stuks* geldt. Dat klopt toch?

PTT: Even in mijn boekje kijken. Ja dat klopt. Maar helaas meneer, Uw pakjes zijn geen belstukken. Ze zijn te klein. Belstukken zijn nu eenmaal stukken waarvoor aangebeld moet worden omdat ze te groot zijn. Belstukken kunnen nu eenmaal niet door de gewone brievenbus. Daarom moet er ook een beetje meer porto op. Regels zijn nu eenmaal regels. Ook bij de PTT.

NTG: Maar meneer dat klopt toch niet helemaal. Volgens Uw tarievenboekje zijn belstukken soms goedkoper te versturen. Ze moeten dan wel in een bepaalde tariefgroep vallen en het minimum aantal moet 25 stuks zijn. Jawel, ik heb Uw tarievenboek zeer goed bestudeerd.

PTT: Laat mij weer eens in mijn boekje kijken. Sapperloot U heeft gelijk. *Maar* toch meneer, Uw pakjes blijven te klein. Dus als belstuk accepteren gaat helaas niet. Voor belstukken geldt nu eenmaal een minimum afmeting van 38x26.5x3.2cm en een maximum afmeting van 100x50x50cm. Daartussen moet Uw pakje dus zitten. Als Uw pakjes niet in onze rode bus kunnen, dan zijn ze te groot en dan heeft U een echt belstuk. Uw pakjes voldoen helaas niet aan de minimale maten. Ze zijn te klein. Ik kan er niets aan doen. Regels zijn nu eenmaal regels. Jaja, ook bij de PTT.

NTG: Maar meneer de PTT, ik hoef toch geen lucht te blazen in de pakjes om ze een centimeter dikker te maken waardoor ik ze goedkoper en gemakkelijker kan versturen?

PTT: Jawel meneer, dat mag U doen. Als Uw pakjes door die lucht dan wat dikker worden, kunnen die niet meer door de brievenbus. Het worden dan echte belstukken. U hoeft dan ook niet meer te likken en U ben goedkoper uit.
U ziet meneer, met lucht ben U bij de PTT goedkoper uit. Hahaha. O sorry meneer. . .

NTG: Helaas meneer de PTT. Mijn adem is ook niet meer wat het vroeger was. Mag ik in plaats van mijn lucht ook deze *gatfrisse ponskaartjes* aan de *linker zijkant van de enveloppe* plakken? Worden de pakjes ook iets groter van. En dan krijgen de geadresseerden gelijk iets om over na te denken. Ziet U meneer, de NTG kent ook een aantal verzamelaars en iets antieks op computergebied zal zeker geapprecieerd worden. Het schijnt dat die oude kaartjes al voor een gulden per stuk worden verhandeld. Ziet U, wij doen graag wat extra voor onze leden.

PTT: Zo'n ponskaartje mag natuurlijk ook meneer. Als Uw pakjes maar wat groter worden. Hoe U dat doet maakt ons niets uit. Al plakt U aan elke enveloppe een stok. Als die maar minimaal 38 cm en maximaal 1 meter is. Wij zullen die stokken, o sorry pakjes, dan gaarne voor U versturen. Wij versturen wel eens meer vlaggen. Wij doen alles, als het maar aan onze regels voldoet.

NTG: Meneer ik lees in Uw tarievenboekje dat ik slechts 25 ponskaartjes hoef aan te plakken. De rest kan dan wel gewoon als *busstuks* meegeleverd worden. Hoeft ik niet al mijn ponskaartjes van de hand te doen. Dat klopt toch wel?

PTT: Dat is vreemd. Even in mijn tarievenboekje kijken. Jawel het staat het. Zal dan wel kloppen. Maar ja regels zijn regels. Zeker bij de PTT.

NTG: Astublieft dan meneer, 30 pakketjes met een ponskaartje. Heeft U voor de zekerheid 5 extra. De andere 100 pakjes blijven onveranderd. Voldoet de NTG nu aan Uw belstukkenprotocol?

PTT: Prachtig meneer. Jaja, het ziet er prachtig uit. Nu kunnen we alles volgens onze PTT regels versturen.

NTG: Meneer de PTT, kunt u trouwens ervoor zorgen dat de aangeplakte ponskaartjes heelhuids bij de geadresseerden aankomen? Het zou niet leuk zijn als U ze er weer van af zou scheuren. Het hoort nu eenmaal bij het pakje. Al die vragen die we anders krijgen van onze NTG leden als ze zien dat aan hun pakje wat aangeplakt had gezeten.

PTT: U kunt op ons vertrouwen meneer. Wij doen ons best. Wij zullen alleen de pakketjes soms open moeten maken om te controleren of er wel allemaal drukwerk in zit. En als het interessant materiaal is, kan het misschien *een dag langer* duren voordat het pakje bij de geadresseerde is. Maar wij als PTT doen natuurlijk ons best.

NTG: Dat is goed meneer. Tot ziens dan maar weer. Tot over een half jaar.

Voldaan dat alles toch nog op tijd (en goedkoop) kon worden afgeleverd en verstuurd, stapte Uw NTG-MAPS editor laat in de namiddag het postkantoor uit. Eindelijk weer eens een avondje om niet *achter* de buis, maar er *voor* te gaan zitten. Eens kijken of het programma 'Ook dat nog' uitgezonden wordt...

MAPS inhoud

En daar ligt weer een dikke MAPS voor u. Hopelijk op tijd...(het is maar afwachten als ik dit stukje schrijf). Natuurlijk weer de nodige bijdragen die reeds elders zijn gepubliceerd. Sommige belangrijke artikelen zijn nu eenmaal herpublicatie dubbel en dwars waard! Maar daarnaast ook weer zeer veel nieuw materiaal uit diverse windhoeken. En alle bijdragen, met uitzondering van deze, van een zeer hoog gehalte[1].

Het centrale thema in deze MAPS is wel *Hypertekst* en alles wat er omheen hangt als bijvoorbeeld HTML, WWW, PDF. Diverse overzichtsbijdragen zijn opgenomen doch ook in de diepte vindt u één en ander in deze MAPS.

Nieuw in deze uitgave, en dat valt op! De artikelen over *TEX en kleur*. En als alles in de drukfase goed gaat (volgende week...), vindt U in deze MAPS ook de nodige kleurenpagina's!
Daarnaast vele andere bijdragen natuurlijk ook weer van onze 'huis-auteurs'[2]. Artikelen over Perfecte tekstverwerkers, PostScript en dissertatieproduktie worden altijd graag opgenomen. Natuurlijk ook weer in deze MAPS enkele 'last minute' bijdragen, waaronder nieuws van de gloednieuwe 4allTEX *dubbel* CD-ROM, een eerste indruk van de evaluatie van het pakket Scientific Word / Workplace en een verslag van de BachoTEX'95 bijeenkomst.

MAPS verwerking

Het materiaal werd weer in vele vormen aangeleverd, *geen plain TEX gelukkig meer!*, wel LATEX209, LATEX2ε (sorry LATEX heet dat...),[3] ASCII, WORD- en PDF files en zelfs op papier. Dit laatste in de vorm van een kopie van een artikel in de Franse taal uit *Cahiers GUTenberg* (het lijfblad van de Franse TEX Gebruikersgroep), welk artikel dank zij onze fenomenale tweetalige zuiderbuur **Philippe Vanoverbeke**, na vele zweetdruppels van hem in onze MAPS overgenomen kon worden.
Verwerking ging voor ongeveer 80% via de antieke LATEX2.09 manier. Werkt altijd. Snel en veilig (jaja, uw MAPS editor is zeer conservatief). Een vijftal bijdragen werden slechts separaat via LATEX2ε afgehandeld. En natuurlijk dankzij de onovertroffen 4allTEX CD-ROM op het inmiddels opgewaardeerde Pentium-90 PC-systeem van uw editor, was het *relatief* een fluitje van een cent.

De andere tweelingbroer van CGL

Eind vorige jaar nog vechtend met een niet nader te noemen 'Perfect' documentverwerkingssysteem, kwam Herman Haverkort (u zal deze naam niet meer gauw vergeten) in aanraking met TEX. Las Knuth's boek in een verloren zondag uit, en maakt nu al artikelen die het niveau van gemiddelde (LA)TEX guru te boven gaan.
U kunt het lezen in een drietal uitgebreide artikelen van hem. Onze Frans Goddijn merkte over zijn artikelen in één van onze vele telefoongesprekken op: 'Deze artikelen laten zien wat een beginnende TEX gebruiker onmiddellijk moet kunnen maken. LATEX hoeft helemaal voor een beginner niet moeilijk te zijn...'[4]. Maar, u zal zeker moeten toegeven, het zijn dotjes van bijdragen. Ik kan mij voorstellen dat het eerder genoemde 'Perfecte' documentverwerkingssysteem wel wat moeite heeft om Herman zijn werk na te doen.
Overigens, Herman Haverkort is dermate druk geweest met LATEX dat zijn studie er bij in dreigde te schieten: voor een zwaar tentamen was nog slechts een maandagmiddag beschikbaar om een informatica-standaardwerk over datastructuren door te nemen (van colleges volgen was het al helemaal niet meer gekomen). We hoeven ons echter niet schuldig te voelen dat die leuke *Haverkort stylefiles* voor ons ten koste gingen van zijn studieresultaat: het werd een 8,5!
Al met al: we kunnen ongetwijfeld heel wat van deze Herman in de toekomst verwachten. Onze Kees van der Laan heeft er nu een zware concurrent bij.

MAPS Awards

MAPS auteurs Phons Bloemen, Sebastian Rahtz en J. Hagen (Pragma) hebben al kennis kunnen maken met de shipout van `box255` vol met gekleurde wiebervormige en rekbare objecten: een stimulans, eerbetoon en een blijk van dank van de MAPS redactie aan een relatief beginnende auteur. En wat blijkt: *deze mensen blijven schrijven voor de MAPS!* En zelfs de vorige MAPS Award winnaar doet dat nu in kleur! Wat die rekbare gekleurde objecten niet allemaal kunnen veroorzaken. Prachtig! We zouden eigenlijk alle auteurs zo'n `box` moeten geven, echter één en ander zou zeker tot logistieke problemen leiden.
De vraag wie de Award dit jaar krijgt uitgereikt, hoeft eigenlijk niet moeilijk voor de redactie te zijn...

[1] Dat zullen niet alle auteurs zeggen! Veelal zijn ze zeer bescheiden, te bescheiden veelal. Ze weten soms nauwelijks hoe hun al dan niet eenvoudige bijdrage door andere NTG leden verslonden worden...

[2] Kees, je weet niet van ophouden. Je moet maar eens per bladzijde gaan betalen...

[3] Al waren sommige LATEX documenten eerder TEX documenten te noemen, inclusief alle mogelijke soorten 'afschuwelijkheden' (zoals een herdefinitie van het \\-commando!

[4] *Echte* beginnende TEX gebruikers: neem deze uitspraak wel met een flinke dot glue.

MAPS-sponsoring

Jawel, de voor u liggende dikke MAPS is mogelijk gemaakt door een aantal NTG donateurs. Grote belangstelling opeens om in de MAPS te staan. Op herhaling: **Addison-Wesley**, nieuwkomers: **Adobe Systems**, **Computercollectief** en **Blue Sky Research**, allemaal bekenden binnen onze TEX wereld. Zij hebben allen *een eigen bladzijde* achterin de MAPS waarin ze onze leden op de hoogte kunnen brengen van hun laatste produkten en diensten. Daarnaast kunnen ze een eigen folder bijsluiten of zelfs CD-ROM's. En dat hebben ze ook gedaan! MAPS ontvangers zullen ongetwijfeld twee CD-ROM's in hun dikke MAPS enveloppe hebben aangetroffen. Bedankt allen! Dankzij u heeft de NTG weer een uitgebreide MAPS aan haar leden kunnen sturen.

Wilt u ook een geheel eigen MAPS pagina? Heeft u ook waardevolle informatie voor onze NTG leden en wilt u meehelpen de kosten van de MAPS te drukken? Neem dan contact op met de MAPS redactie, ook indien u potentiële gegadigden weet!

MAPS goes Electronic

Jaja, zoals u in de vorige MAPS heeft kunnen lezen, staan *alle MAPSen* (zelfs inclusief deze uitgave!), op de bekende 4allTEX CD-ROM. Daarnaast zijn oude uitgaven begin dit jaar ook beschikbaar gekomen op CTAN (Comprehensive TEX Archive Network). U vindt ze in de `usergrps/ntg/maps` en in de `usergrps/digests/maps` directory. Beschikbaar als PostScript files, en dank zij onze sponsor **Adobe Systems**, ook in PDF formaat[5] op de nieuwe 4allTEX CD-ROM.

Overigens, het omzetten van PostScript files naar PDF is zonder enige voorkennis van de benodigde programmatuur (zie het Adobe Acrobat artikel elders in deze MAPS) werkelijk een 'fluitje van een cent'. Binnen enkele minuten was door de MAPS redactie de klus geklaard!

MAPS Specials

Ook op dit terrein was het niet stil. Eind 1994 ontvingen we van Sebastian Rahtz, editor van het zeer waardevolle **Baskerville** tijdschrift, een van hun uitgaven[6]. Door guest editor **Robin Fairbairns** was zeer veel werk gestoken in dit prachtige document. Had eigenlijk al 10 jaar eerder moeten verschijnen!

Natuurlijk een document van veel waarde, niet alleen voor de leden van de UK TEX Gebruikersgroep. Daarom direct, na snelle toestemming van Sebastian en Robin, door de MAPS redactie in MAPS Special 1994 omgedoopt. Alle NTG leden hebben ondertussen een *gratis exemplaar* ontvangen.

MAPS & Baskerville

Nu we het toch over de UK TEX Gebruikersgroep hebben. Afgelopen januari werd door deze groep een bijeenkomst georganiseerd met als thema *'Portable Documents: Acrobat, SGML, and TEX'*[7]. Van mijn werkgever (het *Energieonderzoek Centrum Nederland (ECN)*; die naam mag ook wel eens worden genoemd), kreeg ik toestemming naar deze 'conferentie' te gaan. Met twee keer een nachtboot was ik slechts één werkdag kwijt aan het gehele gebeuren. Gehouden in een typisch Londen's theatergebouw, een kleine 200 deelnemers, goede lezingen, zeer gezellige sfeer en een interessant forum (te lezen natuurlijk in de Baskerville) op het eind. Enkele contacten meteen gelegd: nagenoeg al hun conference bijdragen vindt U in deze MAPS[8]: *Bijlage R t/m W*. Bedankt UKTUGgers!

MAPS #15

Zoals u inmiddels ongetwijfeld reeds heeft vernomen, wordt de volgende EuroTEX bijeenkomst in Nederland georganiseerd. Van 4–8 september op Papendal te Arnhem. Voorlopig is besloten om in het najaar *geen MAPS* uit te brengen in de u bekende vorm.

Daar de NTG 1995 najaarsbijeenkomst samen zal vallen met de EuroTEX bijeenkomst, zullen de EuroTEX'95 proceedings *ook aan alle NTG leden* worden gestuurd. Mogelijk daarnaast (en dat is mede afhankelijk van onze donateurs), zal een zeer dunne MAPS (nu eens een ander uiterste voor de afwisseling) verschijnen met onder meer het verslag van de 1995 voorjaarsbijeenkomst en hooguit enkele 'noodzakelijke' artikelen.

Deze keer vragen wij u daarom dringend *geen bijdragen voor de komende MAPS in te sturen*. Mogelijke potentiële auteurs zullen daarom ook niets van mij horen[9], Begin 1996 mag weer aangeleverd worden. Kan ik deze zomer eindelijk eens rustig met vakantie gaan...

Tot slot[10]

Natuurlijk was de productie van deze MAPS niet mogelijk geweest zonder het vele werk van **Wietse Dol**, **Frans Goddijn** en **Jos Winnink**. Jawel, Frans hebben we ook maar bij het redactieteam betrokken. *Van zeer goede en harde werkers moet men altijd profiteren*!

[5] De Acrobat reader, waarmee PDF (Portable Document Format) files afgebeeld kunnen worden, is te vinden op de CD-ROM die bij deze MAPS is bijgesloten.

[6] En altijd stipt op tijd, zowel electronisch als op papier. Perfect Sebastian!. Overigens een abonnement op de Baskerville is *zeer de moeite waard*. Neem contact op met Peter Abbott, 1 Eymore Close, Selly Oak, Birmingham B29 4LB (`uktug-enquiries@tex.ac.uk`).

[7] Het jaar 1995 lijkt wel het hypertext jaar te zijn: onze Franse collega's organiseerden in januari reeds hun hypertext bijeenkomst; de Franse Cahiers GUTenberg #19 van januari 1995 was speciaal gewijd aan dit onderwerp; de NTG had ruim een jaar geleden al voor de 1995 voorjaarsbijeenkomst het thema *hypertext* gekozen; komende TUG'95 bijeenkomst heeft een speciale hypertext dag, en één van de TUGboats dit jaar wordt speciaal aan dit onderwerp gewijd.

[8] En natuurlijk in de **Baskerville** van maart 1995.

[9] Om tot een MAPS te komen worden zo rond de 100–200 e-mails door mij verstuurd, inclusief alle heen en weer discussies. Ongetwijfeld vergeet ik daarbij sommige personen (snel) te antwoorden. Mijn excuses daarvoor.

[10] Mijn excuses voor al die voetnoten. Ik lijk CGL wel. Of ben ik toch door hem aangestoken?

Concept begroting van de Nederlandstalige TEX Gebruikersgroep

voor het jaar 1995

Wietse Dol

Inkomsten		**Uitgaven**	
Contributie	*f* 21.500,00	Administratie	*f* 800,00
Sponsoring		Kamer van Koophandel	*f* 61,00
Rente	*f* 1.400,00	Bijeenkomsten	
		Bestuurskosten	*f* 400,00
		Computerfaciliteiten	PM
		Nieuwsbrief/Verslagen	*f* 16.200,00
		Reis bijdragen	*f* 2.000,00
		Representatie	*f* 200,00
		Bijdrage Bursary fund	*f* 1.000,00
		Afschrijving	*f* 1.182,00
		Sponsoring	*f* 2.000,00
Begroot verlies	*f* 1.000,00	Onvoorzien	*f* 57,00
	f 23.900,00		*f* 23.900,00

Hierboven vindt U de voorlopige begroting voor 1995 van de Nederlandstalige TEX gebruikersgroep. Een toelichting volgt hieronder.

Toelichting

- **Inkomsten**:
  1. Contributie
     De post contributie is gebaseerd op het aantal betalende leden in oktober 1994 en een verhoging van de contributie met *f* 15,00 per lid (instituten met 45 gulden). De ledenstand van oktober 1994 was:

35	instituten	35	× *f* 200,00	*f* 7.000,00
5	aanvulling	5	× *f* 50,00	*f* 250,00
10	studenten	10	× *f* 50,00	*f* 500,00
82	personen	82	× *f* 75,00	*f* 6.150,00
16	TUG leden	16	× *f* 11,50	*f* 184.00
62	personen	62	× *f* 67,50	*f* 4.185,00
				f 18.269,00

  2. Sponsoring
     Er wordt geen sponsoring verwacht, alhoewel voor de MAPS al enige gegadigden voor advertenties zijn gevonden.
  3. Rente
     De vereniging heeft in de afgelopen jaren een behoorlijk kapitaal opgebouwd. Als dit niet nodig is om een tegenvaller op te vangen is het mogelijk behoorlijk wat rente te krijgen. Daarnaast heeft in 1994 de verkoop van de CD-ROM ervoor gezorgd dat gedurende langere tijd een groot bedrag op de spaarrekening heeft gestaan. De rente op die rekening wordt begin 1995 uitgekeerd.
  4. Begroot verlies
     Naar verwachting zal de verkoop van de CD-ROM in 1995 door gaan en voor de vereniging inkomsten opleveren. Dit kapitaal zal gebruikt worden voor TEX activiteiten (zoals bijvoorbeeld EuroTEX, WWW, CD-ROM projecten etc). Willen we de contributie in 1 jaar niet te veel doen stijgen dan kan met een gerust hart een verlies worden begroot. Dit temeer omdat naar alle waarschijnlijkheid de posten contributie, rente en uitgaven MAPS gunstiger zullen uitvallen dan begroot.

- **Uitgaven**:
  1. Administratie
     Dit is bedoeld voor materiaal voor de secretaris en penningmeester. De hoogte is bepaald aan de hand van de hoogte van de uitkomst over 1993 en de realisatie in 1994 tot nu toe.
  2. Kamer van Koophandel
     Dit is een jaarlijks terugkerende inschrijving van *f* 61,00.
  3. Bestuurskosten
     Hieronder vallen kosten als telefonische vergaderingen, vergoeding reiskosten voor een eventuele physieke bijeenkomst etc.

4. Computerfaciliteiten
   We maken gebruik van fileserver faciliteiten. Die worden op dit moment niet in rekening gebracht, maar dit kan in de toekomst wel eens veranderen. Vandaar dat dit als PM-post wordt opgevoerd.
5. Nieuwsbrief/Verslagen
   Het copiëren en verspreiden van de verslagen van de bijeenkomsten. De kosten bedragen ongeveer *f* 30,00 per exemplaar. Er is rekening gehouden met twee maal een oplage van 300 exemplaren.
6. Reisbijdragen
   Het is de bedoeling dat de vereniging bijdraagt in de kosten van het bijwonen van buitenlandse bijeenkomsten die met TEX te maken hebben. In 1995 moet rekening gehouden met het bijwonen van de TUG bijeenkomst in de VS (oostkust).
   Als tegenprestatie wordt een verslag van de bijeenkomst verwacht, ter publicatie binnen de vereniging.
7. Bijdrage Bursary fund
   In 1993 is besloten jaarlijks een bijdrage te leveren aan het Bursary fund van TEX conferenties. Dit fonds is bedoeld om TEX gebruikers met financiële problemen in de gelegenheid te stellen een internationale TEX bijeenkomst bij te wonen.
8. Afschrijving
   In 1994 is voor het secretariaat een laserprinter aangeschaft. Deze wordt over drie jaar afgeschreven.
9. Sponsoring
   Het bestuur is van plan in 1995 een financiële bijdrage te leveren aan het bulletin board van Frans Goddijn (*f* 1000,00). Ook kan opnieuw geld beschikbaar gesteld worden voor het LATEX3 project.
10. Representatie
    Als bestuursleden van zustervereningingen bij onze bijeenkomst uitgenodigd worden, kan een tegemoetkoming in de kosten worden gegeven.

Financieel verslag 1994
Nederlandstalige TEX Gebruikersgroep

Wietse Dol

April 1995

1 Inleiding

Voor U ligt het financieel jaarverslag van de vereniging NTG over het jaar 1994. De financiële ontwikkeling van de vereniging wordt aan de hand van de winst en verliesrekening en de balans besproken. Om een goede indruk te krijgen van het verschil tussen de begroting en realisatie van 1994 is tevens in tabel 1 de begroting van 1994 opgenomen.

2 De winst en verliesrekening

In tabel 2 is de winst- en verliesrekening over 1994 weergegeven. De diverse posten worden in de volgende paragrafen kort toegelicht.

Bedragen in guldens	**Debet**	**Credit**
Contributie		16.000,00
Sponsoring	3.000,00	
Rente		1.200,00
Evenementen		
Administratie	800,00	
KvK en notaris	61,00	
Bestuurskosten	400,00	
Computer fac.		
Mededelingen a/d leden	10.800,00	
Reisbijdragen	3.500,00	
Representatie	400,00	
Verkoop		
Diversen	239,00	
Tekort		2.000,00
	19.200,00	19.200,00

Tabel 1: *Begroting voor 1994*

2.1 Inkomsten

- **Contributies**
  De NTG voert een actieve ledenwerving en dat heeft net als in voorgaande jaren er weer voor gezorgd dat het ledenaantal (en derhalve het contributie inkomen) fors is gestegen (*f* 2.594,79 t.o.v. de begroting).
- **Rente**
  Ondanks het feit dat de NTG verlies lijdt en de rente percentages aan de lage kant zijn is er een fors bedrag aan rente binnen gekomen. Dit heeft alles te maken met de enorme CD-ROM verkopen.
- **CD-ROM verkoop**
  Door de enorme verkoop cijfers van de 4allTEX CD-ROM lijkt de winst- en verlies rekening veel te positief. Als we ons realiseren dat het netto saldo van de CD-ROM verkopen (inkomsten - kosten - investeringen in het project) *f* 14.187,88 bedraagt, dan is simpel uit te rekening dat de NTG over het boekjaar 1994 een verlies heeft geleden van *f* 2.211,22. De CD-ROM opbrengsten moeten als een meevaller worden beschouwd maar kunnen niet worden gebruikt om systematische tekorten aan te vullen. Tevens zijn de CD-ROM inkomsten geen begrotingspost en heeft het bestuur met het CD-ROM projectteam vaste afspraken hoe dit geld verdeeld en gebruikt kan/mag worden.

Bedragen in guldens	**Debet**	**Credit**
Contributie		18.594,79
Sponsoring	1.800,00	
Rente		1.286,19
Evenementen	392,46	
Administratie	981,28	
KvK en notaris	82,00	
Bestuurskosten	639,06	
Computer fac.		
Mededelingen a/d leden	14.013,21	
Reisbijdragen	2.785,50	
Representatie	47,25	
Verkoop		14.073,43
Diversen	1.237,00	
Saldo	11.976,66	
	33.954,42	33.954,42

Tabel 2: *De winst en verliesrekening over 1994*

2.2 Uitgaven

- **Evenementen**
  Het organiseren van de ledenvergaderingen en de NTG-dagen kost geld (b.v. reiskosten van sprekers en zaalhuur). Ook zijn er twee cursussen georganiseerd. De eerste cursus was in Groningen: een 4TEX cursus gegeven door E. Frambach en W. Dol. Van de opbrengst van deze cursus is een SCSI CD-ROM drive gekocht voor het FGBBS. De tweede cursus was de LATEX2ε cursus van Michel Goossens in Antwerpen. Deze cursus heeft bijna niets gekost omdat de cursusleider tevreden was met een kostenvergoeding.
- **Sponsoring**
  Er is dit jaar geld gegaan naar het bursary fund en naar het FGBBS.

- **Administratie**
  Er is meer geld aan administratie uitgegeven dan begroot. Dit komt doordat er meer (potentiële) leden zijn en er een zeer ondersteunend beleid wordt gevoerd door onze secretaris. Dit kost geld maar is voor het functioneren van de vereniging van essentieel belang.
- **KvK en notaris**
  Deze post omvat de kosten van de inschrijving bij de kamer van koophandel in Groningen.
- **Bestuurskosten**
  Deze post bestaat uit de kosten van telefonische bestuursvergaderingen. In 1994 is ook begonnen aan het lijfelijk vergaderen, hetgeen reiskosten met zich meebrengt maar een prettiger en effectiever manier van vergaderen is.
- **Mededelingen aan de leden**
  Deze post bevat de uitgaven gedaan voor het maken, reproduceren en verzenden van het 'verslag met bijlagen'; kortom de MAPS en alle andere spullen die u op u deurmat hoort ploffen. Er is heel veel meer dan begroot (*f*3.213,21) uitgegeven. Dit heeft alles te maken met de kwaliteit en hoeveelheid van de MAPS, maar ook met het feit dat we veel meer leden hebben dan begroot en dat bijna al het contributiegeld terug komt als MAPS.
- **Reisbijdragen**
  Er zijn reizen gemaakt naar Amerika (TEX Users Group: J. Braams) en naar Polen (EuroTEX: J.Braams, W.Dol, E Frambach). Dit heeft meer geld gekost dan op de begroting staat, maar is mede gedaan om PR redenen en het verkoop van de CD-ROM te stimuleren. De afgelopen jaren speelt de NTG een steeds belangrijker rol in de TEX-wereld, dit heeft tot gevolg dat er meer gereisd moet worden.
- **Representatie**
  Eigenlijk een restpost waar alle soorten PR-beleid onder kunnen vallen. Veel PR-beleid zit echter al in de kosten die de secretaris maakt.
- **Diversen**
  Ook hier een grotere post dan voorzien. Dit bedrag wordt echter veroorzaakt door het royeren van een aantal leden en het afboeken (op deze post) van deze dubieuze debiteuren.
- **Saldo**
  Waarachtig een saldo om van te watertanden. Gelukkig wordt dit bedrag niet veroorzaakt door te zuinig beleid van uw bestuur maar door de CD-ROM inkomsten. Let wel dit jaar is een verlies geleden van *f*2.211,22. Dit is het tweede achtereen volgende jaar dat een fors verlies wordt gelden en motiveert de (gerealiseerde) contributie verhoging voor 1995.

3 De balans

In tabel 3 is de balans per 1 februari 1995 weergegeven en in tabel 4 is de balans per 31 december 1993 weergegeven.

De post 'Contributies' betreft leden die hun contributie voor 1994 niet of onvolledig betaald hebben. Het onvolledig betalen wordt veroorzaakt door het geen rekening houden met transfer provisie bij een betaling vanuit België.

De post 'Debiteuren' betreft geld dat van de TEX Users Group rekening moet worden overgeboekt en enige andere posten die nog betaald moeten worden aan de NTG.

De post 'Crediteuren' betreft geld dat naar de TEX Users Group-rekening moet worden overgeboekt en een paar nog niet betaalde declaraties.

Het kapitaal is in 1994 toegenomen met *f*4.884,36; bij de bespreking van de verlies en winst rekening is al toegelicht dat de CD-ROM de reden van deze toename is.

Bedragen in guldens	**Aktiva**	**Passiva**
Giro	22.095,54	
Kas	532,70	
Contributies	671,00	
Debiteuren	450,94	
Crediteuren		2.403,31
Kapitaal		22.528,93
Apparatuur	2.364,10	
Afschrijving		1.182,05
	26.114,29	26.114,29

Tabel 3: *De balans per 1 februari 1995*

Bedragen in guldens	**Aktiva**	**Passiva**
Giro	14.078,01	
Kas		
Contributies		
Debiteuren	942,20	
Crediteuren		2.103,85
Kapitaal		12.916,36
Apparatuur		
Afschrijving		
	15.020,21	15.020,21

Tabel 4: *De balans per 31 december 1993*

4 Conclusie

De vereniging NTG heeft meer geld uitgegeven dan zij heeft ontvangen in 1994. Aangezien dit het tweede achtereenvolgende jaar met een systematisch verlies is, is terecht op de vorige ledenvergadering besloten om de contributie per 1 januari 1995 te verhogen met *f*15,–.

De CD-ROM verkopen verlopen beter dan verwacht en geven de NTG de nodige financiële armslag om activiteiten te ontplooien. Met de EuroTEX meeting op komst en de extra activiteiten t.a.v. de CD-ROM belooft ook 1995 een prachtig NTG-jaar te worden.

Jaarverslag NTG

jan–dec 1994

Gerard van Nes

1 Algemeen

Het jaar 1994 was voor de NTG een zeer druk doch een zeer succesvol jaar. Topper was natuurlijk het uitbrengen van de 4allTEX CD-ROM. Twee druk bezette NTG bijeenkomsten vonden plaats met vele goede lezingen van sprekers van zowel binnen als buiten de NTG. Er verschenen een tweetal uitgebreide MAPS uitgaven (Minutes & APpendiceS). Een MAPS'94 Special Edition werd gemaakt. Medewerking werd wederom verleend aan het LATEX2ε projekt.
En voor de rest is er het diverse door de leden gedaan waaronder het beheer van het bulletin board FGBBS, het beheer van de TEX-NL en de uitgebreide hulpverlening bij vragen op de nationale TEX-NL en 4TEXdiscussielijst.

2 Het NTG bestuur

Het NTG bestuur bestond uit de volgende personen:

- C.G. van der Laan, voorzitter (tot juni 1994)
- J.L. Braams, penningmeester (tot juni 1994),
- J.L. Braams, voorzitter (vanaf juni 1994)
- G.J.H. van Nes, secretaris,
- W. Dol, penningmeester (vanaf juni 1994),
- E.H.M. Frambach, bestuurslid
- J.J. Winnink, bestuurslid (tot juni 1994),
- F. Goddijn, bestuurslid (vanaf juni 1994).

Op de NTG bijeenkomst in juni vonden de bestuursverkiezingen plaats.

C.G. van der Laan en J.J. Winnink traden af. Ze zouden volledig ten dienste van de NTG blijven in functie van respectievelijk Oost-Europa TEX vertegenwoordiger (guru) en MAPS redactielid. Beiden vormden ook de leden van de (nieuwe) NTG commissie van oudgedienden.
C.G. van der Laan werd vanwege zijn vele verdiensten als mede-oprichter van de NTG en zijn vele werkzaamheden in de afgelopen 6 NTG jaren benoemd tot *erelid van de NTG* en ontving de unieke allereeste 'gold'-CD-ROM namens de gehele vereniging.
De door het bestuur voorgestelde twee kandidaten, W. Dol en F. Goddijn werden bij acclamatie gekozen.
Philippe Vanoverbeke continueerde ook in 1994 zijn functie van NTG België-commissaris en verstevigde daarmee de contacten met onze Vlaamse leden.

3 Het NTG ledenbestand

Het ledenaantal bleef ook weer in 1994 duidelijk stijgen. Eind 1994 telde de NTG 265 leden waarvan 36 instituutsleden. Eind 1990/1991/1992/1993 bedroeg het aantal leden respectievelijk 117/159/193/232 en het aantal instituutsleden 21/28/30/33. Een stijging die mede dankzij MAPS en 4allTEX CD-ROM activiteiten een structureel karakter lijkt te krijgen.
De contributie bleef in 1994 ongewijzigd. Vanwege de verhoogde MAPS- en algemene portokosten werd besloten om de contributie, die sinds de oprichting van de vereniging ongewijzigd was gebleven, in 1995 een weinig te verhogen.
De samenwerking van NTG met TUG, waarbij het lidmaatschap gecombineerd kon worden overgemaakt, bleek te resulteren in 83 NTG/TUG leden eind 1994 (74 eind 1993).
De volledige NTG ledenlijst met aanvullende hardware- en software informatie werd gepubliceerd in de MAPS uitgave van november 1994.

4 De NTG bijeenkomsten

Er zijn in 1994 ook weer twee NTG bijeenkomsten georganiseerd welke gekenmerkt werden door een levendige discussie en een hoge mate van informatie-uitwisseling.

1. Op 9 juni 1994 bij de Rijksuniversiteit te Groningen.
   Aanwezig waren 40 leden.
2. Op 17 november 1994 bij Universitaire Instelling Antwerpen.
   Aanwezig waren 55 leden.

Op deze bijeenkomsten, met als thema respectievelijk: 'TEX/LATEX-, METAFONT-gereedschappen' en 'TEX en Publiceren' vonden de volgende lezingen plaats:

- *'Fonts'*, door Erik-Jan Vens;
- *'Gebruik en Management van TEX in een Unix omgeving'*, door Piet van Oostrum (RUU);
- *'LATEX2e: Standard Document Classes and Packages; Upgrading old styles*', door Johannes Braams (PTT Research Lab.);
- *'Manmac BLUes*', door Kees van der Laan;
- *'Electronisch publiceren bij Elsevier'*, door Simon Pepping (Elsevier Science);
- *'Electronisch publiceren bij Kluwer'*, door Rob de Jeu (Kluwer);
- *'Het maken van een etymologisch woordenboek met TEX*', door Prof. H. Blume;
- *'Real Life Book Production: ervaringen met "'The LATEX Companion"'*, door Michel Goossens (CERN);
- *'Maken van IJslands boek'*, door Andrea de Leeuw van Weenen (RUL);

Van bovengenoemde twee vergaderingen (en lezingen) verschenen verslagen met bijlagen (zie MAPS 94.1 en MAPS 94.2).

Een deel van de lezingen werden gepresenteerd door sprekers, die door de NTG speciaal voor deze gelegenheden waren uitgenodigd.
En natuurlijk kwam ook het nodig over 4TEX (en 4allTEX) weer ter sprake met als vaste sprekers Wietse Dol en Erik Frambach.
De leestafels met TEX-achtige boeken, alle vorige uitgaven van de MAPS, diverse LUG- en andere tijdschriften, brochures en diverse aan TEX verwante documenten, trok zoals altijd weer de nodige belangstelling. Bij zowel de voorjaars- als de najaarsbijeenkomst konden de aanwezigen Addison-Wesley (LA)TEX/PostScript boeken met reductie aanschaffen.

5 De NTG MAPS

Ook in 1994 verschenen er twee MAPS uitgaven met ongeveer 410 (!!) goed gevulde bladzijden (in 1992 en 1993: respectievelijk ongeveer 340 en 475). Daarbij onder meer als inhoud: verslagen van nationale en internationale bijeenkomsten, book- en software reviews, artikelen m.b.t. de NTG lezingen, zeer veel tutorial- en ander educatiemateriaal, TEX/LATEX & PostScript bijdragen, font artikelen, TEX en PC's, hypertext, TEX en LATEX macro's, de laatste informatie over LATEX3, algemene- en technische TEX en LATEX bijdragen.
De MAPS redactie bestond in 1994 uit Wietse Dol, Gerard van Nes en Jos Winnink.
Alle MAPS uitgaven werden ook beschikbaar gesteld aan de 4allTEX CD-ROM. Daarnaast werd eind 1994 oude uitgaven op CTAN geplaatst.

Ook werd nog net op het eind van het jaar een MAPS'94 Special Edition gemaakt. Basis hiervoor waren de 100 'Frequently Asked Questions' uit het tijdschrift van de UK TEX Gebruikersgroep 'Baskerville'. De versturing aan de NTG leden heeft begin 1995 plaatsgevonden.

6 De NTG werkgroepen

De functie van de NTG werkgroepen blijkt toch wat op zijn retour te zijn. Bijdragen komen hoofdzakelijk van individuele leden. De veel gebruikte `TEX-NL` discussielijst zorgt dikwijls voor een vorm van 'overleg'.

7 De NTG CD-ROM

De klapper van 1994. Een half jaar na het idee, werd de *eerste 4allTEX* CD-ROM naar de grondlegger van het gehele gebeuren — *Eberhard Mattes* — toegestuurd. De eerste 'gold-disc' werd op de voorjaarsbijeenkomst overhandigd aan de vertrekkende voorzitter.
Gemaakt in een oplage van 600 stuks, bleek deze CD-ROM al binnen 2 maanden volledig te zijn uitverkocht. Begin september kwam een tweede versie beschikbaar. Eind 1994 waren ongeveer 1300 CD-ROM's in andere handen overgegaan.
Veel werk werd hiertoe verzet door het koppel Wietse Dol & Erik Frambach, doch ook Phons Bloemen en diverse mensen er omheen hebben het nodige bijgedragen aan het succes.
Dat 1994 slechts een voorproefje zou zijn van hetgeen in 1995 zou gaan gebeuren, had niemand in het begin van 1994 maar enigszins kunnen indenken. . .

8 NTG op World Wide Web

Phons Bloemen en Henk Haan zijn druk bezig geweest met het maken van NTG-WWW pagina's. Een *experimentele versie* is reeds beschikbaar op `http://ei0.ei.ele.tue.nl/~phons/ntg/ntg.html`.
Acties zijn door Jules van Weerden gestart om een aparte `www.ntg.nl` WWW-site te kunnen openen.

9 Bestuursactiviteiten

- De voorzitter had vanuit zijn functie zitting in de Board of Directors van TUG.
- De NTG wisselde informatie uit met de LUGs: Exemplaren van de MAPS werden verstuurd naar de meeste LUG's. Van een aantal LUG's ontving de NTG de periodieken (zie ook leestafel NTG bijeenkomsten!).
- Namens de NTG werden in 1994 de volgende bijeenkomsten bezocht:
  - TUG'94 te Santa Barbara, Californië.
    Van 31 juli tot 4 augustus werd door de voorzitter deelgenomen aan deze jaarlijkse TUG meeting. Een lezing werd door hem aldaar gegeven.
  - EuroTeX'94 te Gdańsk, Polen.
    Van 25 tot 30 september werd door de voorzitter, de beide leden van het 4TEX team en de oud-voorzitter Kees van der Laan, deelgenomen aan deze Poolse TEX gebruikersbijeenkomst. Zowel een aantal lezingen als een 4TEX- en een Manmac cursus namen deze NTG deelnemers voor hun rekening.
  - CyrTUG'94 te Dubna, Rusland.
    Van 7 tot 10 september werd door Kees van der Laan deze Oost-Europese bijeenkomst bijgewooond. Gekoppeld aan deze trip werd tevens een bezoek gebracht aan Kazan en Petersburg waarbij voordrachten gegeven werden aan de locale CyrTUG gemeenschap.
  - BachoTEX'94 te Polen.
    Van 30 april tot 3 mei werd eveneens door de toenmalige voorzitter deze Oost-Europese GUST'94 bijeenkomst bijgewoond. Lezing over BLUe's zaken werd gegeven.
  - NTUG'94 te Noorwegen.
    Gehouden op 16 mei en bijgewoond eveneens door de toenmalige voorzitter.

  Van de TUG'94 en de CyrTUG'94 meeting (inclusief rondreis) is uitgebreid verslag gedaan in MAPS 94.2. De BachoTeX'94 en de NTUG'94 bijeenkomst werden verslagen in MAPS 94.1.
  Een aantal van bovengenoemde bijeenkomsten werden op uitnodiging bijgewoond.

- Op 3 maart, 22 juni en 10 november vonden telefonische bestuursvergaderingen plaats waarin naast de algemene gang van zaken binnen de NTG, onder meer de organisatie van de NTG bijeenkomsten, Public Relation activiteiten, 4allTEX (distributiemogelijkheden), contributievaststelling en de gebeurtenissen binnen TUG en andere zusterorganisaties ter sprake kwamen. Op 16 september vond een fysieke bijeenkomst bij de voorzitter thuis plaats.

10 Diversen

- Van zowel de TEX-NL fileserver als de TEX-NL listserver werd in 1994 zoals wederom veel gebruik gemaakt. Het aantal abonnees van de listserver bleef tamelijk stabiel gedurende het gehele jaar. Voor respectievelijk eind 1992/1993/1994: 162, 180 en 175. Gedurende respectievelijk 1992/1993/1994 werden 1097/1047/1242 e-mails op TEX-NL verstuurd. Boven aan de lijst Piet van Oostrum die met 10% van de e-mails veel TEX gebruikers uitstekend wist te ondersteunen.
  De activiteiten van de TEX-NL fileserver waren daarentegen minimaal, vooral vanwege de vele andere alternatieven.
- Een nieuwe discussielijst was die van 4TEX. Gestart in april 1994, bleek deze lijst met 170 abonnees eind december, duidelijk in een behoefte te voorzien.
  Er werden in de 304 dagen van het bestaan in het totaal 572 e-mails verstuurd met als top-3 lijst van verstuurders: Wietse Dol, Erik Frambach en Frans Goddijn. Ze namen ieder ongeveer 15% van de e-mails voor hun rekening.
- Het NTG-FGBBS Bulletin Board van Frans Goddijn (en aandrager Henk de Haan) werd door zowel NTG-ers als niet NTG-ers dagelijks bezocht. Het FGBBS bevat op dit moment een indrukwekkende hoeveelheid (LA)TEX materiaal.
- Vele leden maakten via e-mail dan wel via ftp dankbaar gebruik van de RUU fileserver van Piet van Oostrum voor het verkrijgen van diverse soorten TEX materiaal. Daarnaast werd ook gebruik gemaakt van enkele andere ftp-server systemen in Nederland.
- Actieve medewerking werd verleend aan het LATEX2$_\varepsilon$ project door het LATEX3 teamlid Johannes Braams.
- Door Johannes Braams werden zowel op TUG'94 als EuroTEX'94 lezingen m.b.t. LATEX2$_\varepsilon$ gegeven. Wietse Dol en Erik Frambach toonden op de EuroTEX'94 conferentie de mogelijkheden van 4TEX zowel in de vorm van lezingen als in de vorm van een cursus.

From the TUG President

Christina Thiele

April 20th, 1995

TUG and the NTG

I would like to thank Gerard van Nes for offering me this page to write about TUG and our activities. Two years ago, Kees van der Laan had approached me to write some words of greetings on the occasion of the NTG's 5th anniversary, the Lustrum. At that time, I wrote about the good relations which existed between TUG and the NTG; indeed, good relations amongst all user groups are critical to our continuance. We are all essentially volunteer organisations; while some have paid staff in order to deal with administrative issues, the main thrust of each group comes from its volunteers, particularly those who offer to work on committees and boards.

A volunteer 'job' is quite different from other jobs. We do things because we want to; we work late because we want to; we help people, give advice, share coding secrets and shortcuts — because we want to. It's a matter of good will. What we also find, as volunteers, is that goodwill can sometimes be derailed by misunderstandings, by a lack of communication, by an absence of recognition of our efforts. When you add to that mix factors such as e-mail (great, but it still lacks that human quality not even a smiley `;-)` can bring to the screen), and a few language transitions, then the potential for difficulties must be recognised.

And occasionally there have been some hiccups in the relations between our two groups. Last year saw some confusion about TUG's support of the NTG's 4AllTEX CD (still the first CD with TEX ready to go); there was never any question of TUG's support for the idea or the necessity of having such a CD available. However, we were not in a position to offer financial support (subsidising the CD for conference attendees, or selling it directly from the TUG office). Nevertheless, we *have* been able to offer support in other ways. CD orders can be made directly to TUG for forwarding to the NTG; information about the latest editions appears in TTN; and we are hoping to even see a 4AllTEX demonstration at the upcoming TUG meeting in Florida.

What the NTG has offered the entire TEX community in its support for the production of the CD is a major achievement. What other groups can offer is support in whatever form is available to them.

Where's TUGboat?

'*TUGboat* is delayed.' You've been seeing that phrase now for several months. And as difficult as it is to say, it is necessary to say yet again: '*TUGboat* is delayed.' The delays, however, are not easily solved. We have depended on one person for 15 years to edit and produce the issue; one person and one site. And while I had wanted to see a gradual transition away from both of these sole-source dependencies, that has not happened. It may well be that there will be a serious revamping of *TUGboat* in the coming year. For now, all we can ask of our members is that they continue to be patient. TUG does not have the resources to hire a team of production specialists to get the issues out. And so we are facing the other side of volunteerism: other obligations have to take precedence over what we want to do. TUG and the *TUGboat* editor are now in just such a corner, with very little room to move. We can offer our apologies, our sincere promises to ensure that all issues will eventually arrive, but that is small comfort to someone who wants to get *TUGboat*. We are all very aware of this — and none more so than our beleaguered editor.

TUG elections in progress

The TUG elections are drawing to a close as you read this. There were five board positions open, as well as that of president, the position I have held for almost two and a half years. There were four candidates for the board positions; when this happens, the candidates become board members 'by acclamation' (there is no election necessary). They are Barbara Beeton (returning board member), Karl Berry, Judy Johnston, and Jiří Zlatuška.

There are two candidates for president — Michel Goossens and Mimi Jett — the results will be known in late May/early June.

TUG'95 in Florida — July 24–28

Conference materials are now available via CTAN in `tex-archive/usergrps/tug` in `.ps` and `.asc` formats — look for the `tug95*` files. The conference has a different slant this year: mainly papers presented in the mornings, and demonstrations and workshops mainly in the afternoons. This should provide attendees with a much broader range of information and experience which they can more quickly use back at their places of work — and play! And it's often more fun to see someone using TEX than to hear or read about it! Pick up a copy of the preliminary program from CTAN and see what we mean.

And two wishes . . .

If I could squeeze in just a bit more here . . . I'd like to wish the NTG best of luck with the organizing of EuroTEX'95 in Arnhem this September. And I hope that we continue to work together, keeping one another informed of our activities and initiatives, sharing experiences and information.

Op FGBBS vaart alles wel

tweemaal zo snel

Frans Goddijn

goddijn@fgbbs.iaf.nl

March 1995

Abstract

FGBBS kreeg weer de nodige uitbreidingen, waarvan een sneller modem het meest merkbaar is.

Groter

Henk de Haan[1] zorgt met grote regelmaat voor de toevoer van nieuwe files voor FGBBS. Bijna dagelijks kijkt hij vanaf zijn woonplaats Delft rond op het Internet en verzamelt zo, al `ftp`-end alle nieuwe TEX software die voor ons van belang kan zijn. Wanneer hij op deze manier acht of tien floppies kan vullen, of wanneer een nieuwe β-versie van emTEX is verschenen, gaan die in een enveloppe naar Arnhem waar de files op FGBBS worden geplaatst. Soms betreft het files die reeds aanwezige bestanden vervangen, vaak zijn het nieuwe files die er bij komen. De beschikbare harddisk ruimte nam daardoor af, en we besloten een nieuwe disk van 425 MB aan te schaffen... waarvan inmiddels ruim 300 MB in gebruik is door de programmatuur van FGBBS en de file-bibliotheek. Tegelijk werd de interne RAM geheugencapaciteit van de FGBBS computer uitgebreid naar 16 MB. Daardoor, en door installatie van een nieuwere versie van OS/2, WARP 3 kon FGBBS nog weer wat sneller werken, ook als de sysop voor zichzelf aan het LATEXen was. Om de data-overdracht te versnellen werd onlangs een nieuw modem aangeschaft, een AMIGO modem die kan werken op 28k8 bps. Dat is *tweemaal* zo snel als de vorige modem, die bijna *driemaal* zo duur was!

Gebruik

Hans Eichbaum heeft een programma geschreven waarmee bezoekers van FGBBS een fraai, grafisch beeld te zien krijgen met allerlei gegevens. Eichbaums programma levert ook een tekstversie van dit overzicht:

```
Callhist v.1.07-A2, Statistic Processor
Registered to Frans Goddijn
Statistics file created at Friday, 21-04-1995
Statistics over the last 880 callers.

Most Called hours
Between 00h-01h :   54 (6.14 %)
Between 01h-02h :   21 (2.39 %)
Between 02h-03h :   16 (1.82 %)
Between 03h-04h :    3 (0.34 %)
Between 04h-05h :    1 (0.11 %)
Between 05h-06h :    0 (0.00 %)
Between 06h-07h :    5 (0.57 %)
Between 07h-08h :    6 (0.68 %)
Between 08h-09h :    8 (0.91 %)
Between 09h-10h :   17 (1.93 %)
Between 10h-11h :   28 (3.18 %)
Between 11h-12h :   29 (3.30 %)
Between 12h-13h :   38 (4.32 %)
Between 13h-14h :   41 (4.66 %)
Between 14h-15h :   69 (7.84 %)
Between 15h-16h :   49 (5.57 %)
Between 16h-17h :   44 (5.00 %)
Between 17h-18h :   47 (5.34 %)
Between 18h-19h :   60 (6.82 %)
Between 19h-20h :   64 (7.27 %)
Between 20h-21h :   55 (6.25 %)
Between 21h-22h :   85 (9.66 %)
Between 22h-23h :   74 (8.41 %)
Between 23h-24h :   66 (7.50 %)

Baudrate Statistics
at 28800 Bps (1.48 %)
at 24000 Bps (0.34 %)
at 19200 Bps (0.45 %)
at 14400 Bps (72.05 %)
at  9600 Bps (3.07 %)
at  2400 Bps (22.27 %)

Average of 220 days : 4.00
Statistics from 09-13-94 to 04/21/95 (220 days)
```

Hieruit blijkt dat veruit de meeste bellers met 1400 baud modems werken terwijl een opvallend grote groep nog op 2400 baud communiceert. Een minderheid zit nog met 9600 baud faxmodems die maar niet stuk willen en sommige bellers met echt snelle modems lukt het niet om op de volle snelheid van 28800 baud een verbinding tot stand te brengen. Men belt overdag het meest rond de klok van drie uur en 's avonds het meest rond de klok van tien.

In dit overzicht zijn niet de 'vaste klanten' FGBBS opgenomen, de mensen die met behulp van het TERMINATE programma door de 'achterdeur' van FGBBS alle gewenste bestanden en berichten uitwisselen. Op dit moment zijn dat Henk de Haan, Wietse Dol, Bert Doppenberg, Herman Haverkort, Gerard van Nes, John Timmerman en, vanuit Frankrijk, Philippe Vanoverbeke (alsmede een aantal persoonlijke vrienden van de systeembeheerder). Sommigen van hen maken meermalen per dag contact, anderen eenmaal per dag of per week.

[1] haan@fgbbs.iaf.nl

Herman Haverkort TEX Development Lab

Herman Haverkort, auteur van enkele artikelen in deze MAPS, heeft een eigen 'werkhoek' op FGBBS waar men telkens de meest recente versies kan vinden van de door hem ontwikkelde style files met bijbehorende handleidingen. Het ligt in de bedoeling dat af en toe hiervan een kopie op `ctan` wordt geplaatst, maar voor het allernieuwste loont het de moeite even op FGBBS te kijken.

Toekomst

Hoewel ik door middel van de *information superhighway* programmatuur van OS/2 zo af en toe weleens op het Internet rondzwerf, zijn de kosten hiervan toch zodanig dat dit geen verstandige methode is om intensief te gaan gebruiken voor het *up to date* houden van FGBBS. Het is daarom vooral aan Henk de Haan te danken dat FGBBS bij de tijd blijft. En bovendien: het is één ding om rond te kunnen snuffelen in verre TEX archieven, het is een ander ding om te kunnen oordelen wat daarbij van belang is voor bellers van FGBBS. Wanneer vanaf deze zomer elke Nederlander die niet te ver verwijderd is van een middelgrote stad op lokaal tarief kan gaan internetten, zal het gebruik van FGBBS misschien af gaan nemen, maar het blijft voorlopig de enige plek waar je zo geconcentreerd en gebruikersvriendelijk zo veel TEX bij elkaar kunt vinden.

TEX-NL discussielijst

Jules van Weerden,

(beheerder TEX-NL)

jules.vanweerden@let.ruu.nl

Een kort verhaaltje over de TEX-NL lijst, gezien van uit het beheer.

Toen Evert Jan Evers mij ongeveer twee jaar geleden vroeg om de TEX-NL te gaan beheren, was natuurlijk mijn eerste vraag: hoeveel werk is dat? Ach dat viel wel mee. Drie of vier keer per jaar een berichtje en dat was het wel.

Nou, ik heb het geweten: Om de één of andere reden stond de 'owner' van de lijst niet bovenaan en kreeg Evert Jan geen enkele foutmelding terug. Toen we mij als medeeigenaar toevoegden, werd ik dus *wel* bovenaan gezet. De eerste weken heb ik de 'rommel' van afgelopen maanden mogen opruimen: mensen die allang geen account meer hadden verwijderd en volledige email-adressen verwijderd. Maar nu is het redelijk rustig. Je ziet wel welke machines of instituten regelmatig iets doen met hun apparatuur. Erik Jan Vens zet geloof ik elk weekend zijn machine uit.[1]

Soms mag ik een gebruikerster weggooien en onder een andere (betere?) naam weer toevoegen en soms verdwijnen mensen omdat de LISTSERV machine het adres niet meer terug kan vinden. Ze worden dan ge-auto-delete-ed (is dit nog Nederlands?). Dan wordt het echt detective werk, want ook ik kan dan soms geen mailtje meer sturen om ze te vertellen dat ze er afgegooid zijn. Dat moet dan per post of telefoon. Maar erg vaak gebeurt het niet.

Als het lukt zal ik een klein grafiekje meesturen waarin het verloop van het aantal abonnees op de lijst wordt weergegeven. Er zit een hele kleine stijging in het aantal leden: van 167 in de eerste week van Januari tot 183 vorige week (27 maart 1995).

Nu over de praktische kant van de zaak:

Om *op* de TEX-NL lijst te komen kun je een bericht sturen aan:

```
listserv@nic.surfnet.nl
```

zonder subject (mag wel, maar er wordt niet naar gekeken) en als inhoud:

```
subscribe tex-nl <jouw naam in minimaal
                                  twee woorden>
```

Wil je er weer af stuur dan een bericht naar het zelfde adres met als inhoud:

```
unsubscribe tex-nl
```

Wil je zelf een copy hebben van een bericht dat je naar de lijst hebt gestuurd, stuur de inhoud:

```
set tex-nl repro
```

Wil je statistieken van de lijst hebben geef:

```
stats tex-nl
```

Wil je een lijst van gebruikers (ruim 180): geef

```
review tex-nl
```

Als je gebruik maakt van een 'signature', zet dan op de regel(s) na het commando het woord 'end'. De LISTSERVER zal dan stoppen met zoeken naar andere commando's.

Voor een volledig overzicht van de commando's die je aan de LISTSERVER kunt geven kun je geven:

```
info refcard
```

Als er problemen zijn: stuur een email-bericht naar
`owner-tex-nl@nic.surfnet.nl`
en het beheer (ik dus voorlopig) zal er naar kijken.

Ik bewaar de berichten ook. Op datum en tijd van aankomst. Zie `ftp.let.ruu.nl:/pub/tex/tex-nl` (overigens met de zelfde moeite ook de boodschappen van de 4TEX-lijst).

Met een WWW-lezer als NETSCAPE of MOSAIC kun je ook berichten opzoeken van de TEX-lijst.
`URL: gopher://hearn.nic.surfnet/nl:70/11/`
`surfnet.dir.dutch`.

[1] Heel goed voor 't milieu en het levendig houden van mijn postbus. . .

De NTG op het World Wide Web

Henk de Haan

Ghanastraat 1
2622 GJ Delft
haan@fgbbs.iaf.nl

17 april 1995

Abstract

Het internet is een van de meest snel groeiende elektronische media. Dit is mede te danken aan het zogenaamde *World Wide Web* (WWW). In dit artikel wordt ingegaan op de functie van het Word Wide Web, het gebruikte tekstformaat (HTML) en de mogelijkheden die het voor de NTG kan hebben. Er wordt ingegaan op de huidige (experimentele) service en er worden een aantal mogelijke nieuwe toepassingen vermeld.

Keywords: NTG, Internet, Word Wide Web

Een belangrijk kenmerk van de jaren 90 is de opkomst van interactieve elektronische media. Een belangrijk gedeelte van de aandacht voor deze media is te danken aan het internet, een wereldwijd netwerk waarmee computers over de hele wereld met elkaar verbonden zijn.

Het Internet

Het internet is ontstaan als een netwerk tussen een aantal (Amerikaanse) universiteiten en militaire instellingen, maar het is uitgegroeid tot een wereldomvattend geheel, waaraan ook allerlei bedrijven, non-profit organisaties en particulieren meedoen. Dat laatste is mede te danken aan de aandacht die het internet krijgt in de traditionele media (zoals kranten en televisie).

Het eigenlijke internet bestaat echter alleen maar uit een hoop computers en kabels waarmee elektronische data kan worden getransporteerd. Van oudsher wordt het internet gebruikt voor diensten als elektronische post (E-Mail), discussiegroepen (News), bestandsoverdracht (FTP) en interactieve toegang tot andere computers (Telnet).

De aantrekkingskracht van het internet is voor een groot deel te danken aan het feit dat het niet echt door een centraal orgaan wordt bestuurd, waardoor er genoeg ruimte is voor nieuwe initiatieven. Dit biedt ruimte aan nieuwe aanbieders (waaronder de NTG!) en aan nieuwe dienstvormen.

Het World Wide Web

Als belangrijkste voorbeeld van de nieuwe dienstvormen kan het *World Wide Web* (afkorting: WWW) genoemd worden. Dit is een hypertekst-achtig systeem, waarmee tekstinformatie (met de bijbehorende illustraties) op een grafische manier getoond kan worden. Het World Wide Web wordt in principe niet begrensd door systeembeperkingen (er is programmatuur voor allerlei systemen beschikbaar) en lokaties.[1]

Het World Wide Web biedt gebruikers op computernetwerken een consistente en eenvoudige methode om een hele reeks media te benaderen. Dit gebeurt met een zogenaamde *Browser* waarmee op een lokale Computer documenten op het netwerk bekeken kunnen worden. Er zijn browsers beschikbaar voor allerlei soorten systemen, bijvoorbeeld: *Lynx* (UNIX, VMS), *NCSA Mosaic* (X11/Motif, MS-Windows, Macintosh), *Netscape* (X11, MS-Windows, Macintosh), *Chimera* (X11), *Cello* (MS-Windows) en de *IBM WebExplorer* (OS/2).

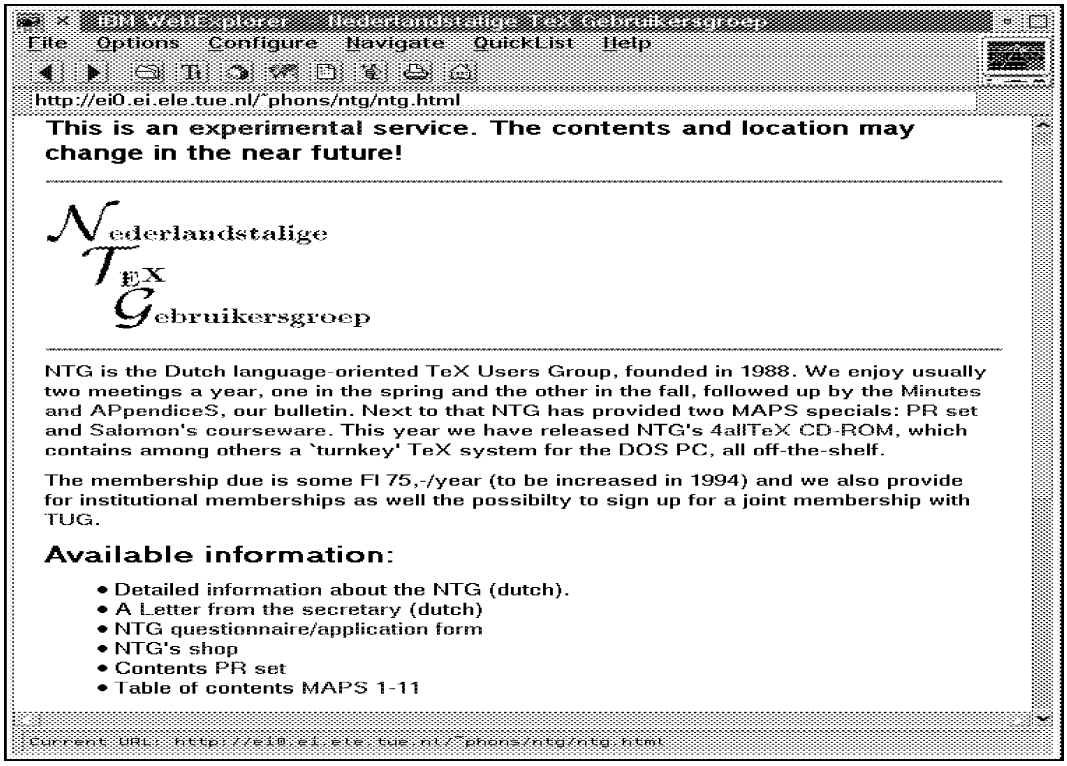
Met behulp van een browser en een internet verbinding worden via het zogenaamde `http` (HyperText Transport Protocol) document van een server gehaald. Deze documenten worden meestal geschreven in de zogenaamde *HTML* taal (HyperText Markup Language). HTML is een op SGML gebaseerde taal waarin de logische structuur van het document voorop staat. Momenteel wordt meestal HTML versie 2 gebruikt, maar er wordt al druk gewerkt aan versie 3. Verder zullen een aantal browsers in de toekomst het Adobe PDF formaat ondersteunen.

Het World Wide Web zal voor veel nieuwe internetgebruikers de primaire interactieve interface worden, terwijl ook veel bestaande internetgebruikers de mogelijkheden van *'het Web'* zullen waarderen. Het World Wide Web zal voor veel bedrijven en organisaties mogelijkheden bieden om informatie voor een groter publiek beschikbaar te maken.

De NTG op het WWW?

Het World Wide Web speelt ook al een rol in de internationale TEX gemeenschap. Zo is het CTAN archief al een tijdje

[1] Een officiële omschrijving van het World Wide Web is: 'a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents'.



Figuur 1: *Openingsscherm van de experimentele NTG-pagina's op het World Wide Web*

benaderbaar,[2] is er online documentatie voor LaTeX en zijn er een aantal TeX gebruikersgroepen met eigen pagina's. Natuurlijk kan de NTG op dit gebied niet achterblijven. In de laatste MAPS (#13, blz. 5) stond een oproep van ondergetekende en tijdens de NTG-bijeenkomst in Antwerpen is ook het een en ander besproken.

Dankzij de inspanningen van een paar individuele NTG leden (Phons Bloemen en ondergetekende, Henk de Haan) zijn er intussen een aantal testpagina's beschikbaar. Deze pagina's zijn **voorlopig** ondergebracht op de URL (Universal Resource Locator) `http://ei0.ei.ele.tue.nl/~phons/ntg/ntg.html` (zie figuur 1).

De pagina's bevatten algemene informatie over de NTG en een aantal van haar activiteiten. Zo is er een informatie over de 4allTeX CD-ROM, de NTG Shop, de NTG PR-Set en de MAPS te vinden. Dit is allemaal informatie die ook via andere kanalen te verkrijgen is, maar die hier op een nette manier, inclusief allerlei kruisverwijzingen, gepresenteerd wordt. Verder zijn er een aantal referenties naar andere relevante TeX adressen op het World Wide Web aanwezig (b.v. een verwijzing naar de EuroTeX pagina). Bij het ontwerp van de pagina's is zoveel mogelijk rekening gehouden met trage links (i.e. grote inline plaatjes zijn niet opgenomen) en verder is het gebruik van browser specifieke features zoveel mogelijk vermeden.

De Toekomst

Alhoewel de NTG pagina's nog niet uitgebreid zijn aangekondigd zijn er toch al vrij veel mensen die deze pagina's ontdekt hebben. Een verdere uitbouw van de NTG-site is mede afhankelijk van een meer permanente locatie. Mogelijke uitbreidingen zijn:

- Interactief aanmeldingsformulier.
- Beschikbaarheid informatie in twee talen (Engels en Nederlands).
- Uitgebreidere informatie over de 4allTeX CD-ROM (inclusief screenshots en bestelmogelijkheden).
- Informatie over individuele (bestuurs)leden.
- Distributie NTG-styles
- Betere MAPS index, waarmee direct postscript bestanden opgehaald kunnen worden.

Natuurlijk zijn er nog veel meer uitbreidingen mogelijk. Suggesties hiervoor zijn welkom bij de Internet Werkgroep.

De Internet Werkgroep

Intussen is er een (informele) internet werkgroep binnen de NTG opgericht. Deze werkgroep bestaat voorlopig uit Phons Bloemen, Jules van Weerden en Henk de Haan. Er wordt momenteel hard gewerkt aan een eigen domeinnaam voor de NTG (`NTG.NL`) en er zijn concrete plannen om een permanente World Wide Web service op te zetten. Tijdens de NTG-voorjaarsbijeenkomst in Enschede zal hierover ongetwijfeld meer informatie beschikbaar zijn!

[2] Bijvoorbeeld `http://jasper.ora.com/ctan.html|`, `http://www.ucc.ie/cgi-bin/ctan` en `http://www.tex.ac.uk/tex-archive/`.

General information 3rd edition of the 4allTEX CD-ROM

May 1th, 1995

Below you will find the most important items concerning the 3rd edition of the 4allTEX CD-ROM.

1 Availability

The first 4allTEX CD-ROM was introduced on June 9th 1994 at the 13th local NTG (Dutch TEX Users Group) meeting in the Netherlands. Within 3 months all 600 copies were sold (all over the world) and the NTG decided to make a second edition that was introduced on September 15th. On this second edition NTG puts an extra 130 Mb in addition to the first release. In march 1995 the NTG sold the last copy of the second edition (i.e. 850 copies of the second release were sold). Therefore the NTG planned a third edition... At the end of May the NTG should be able to distribute the third edition and of course there are even more Megabytes on the third than on the second release. The NTG discovered that all the nice stuff would not fit onto one CD-ROM, so it became a double CD-ROM. This means more than one Gigabyte of TEX related stuff.

2 Contents

4TEX is based on the famous, fast, and well-known emTEX distribution. Besides that, the 4allTEX (yes for all!) CD-ROM contains all related software as available as of April 1995.
So:

1. It contains the latest 4TEX version (3.27; network drive independent). 4TEX gives the user a user friendly interface with a large set of utilities.
2. 4TEX contains software in direct executable form.
3. 4TEX supports more than 60 formats including huge and big versions, TEX, LATEX, Cyrillic, Polish, Greek, TEXinfo, LOLLIPOP, LATEX2ε (official december 1994 release, patch level 3) etc.
4. 4TEX support many printer drivers (including colour deskjets, linotronics, matrix printers, 300/600 dpi laserprinters, laserjets, fax etc).
5. 4TEX supports graphics: both for previewing and printing. All functionality of e.g. BM2FONT, GHOSTSCRIPT 3.12, HP2XX, TEXCAD, QFIG are included, ready to use!
6. 4TEX includes AMSPELL: a spell-checker for languages English (UK), English (USA), Dutch, German, French, Spanish and Italian.
7. 4TEX contains conversion utilities: WP to LATEX/TEX, MS-WORD/CHIWRITER/PC-WRITE/TROFF to LATEX, DETEX, UNTEX, etc.
8. 4allTEX contains a huge set of fonts: why waste your time generating fonts (fax, 300dpi, 600dpi, etc)?
   All fonts are available in .pk format and are not stored into fontlibraries, i.e. they can be used also for non-DOS systems!
9. 4allTEX contains a huge set of .STY .MF .PK .TFM .BST .etc files.
10. 4TEX includes METAFONT with automatic font generation.
11. 4TEX fully supports PostScript fonts. E.g. with automatic font generation using PS2PK! and with the utility MAKEFONT you can convert any PostScript font for use within TEX.
12. With 4TEX you can view and print directly in e.g. Times.
13. 4TEX contains dbase utilities, supports MAKEINDEX, etc.
14. 4TEX uses default the integrated QEDIT 3.0 editor (you can change that in the 4TEX configuration file: use your favourite editor!). With the QEDIT editor you can even do block compilation: mark a small part of your (large) document, and *only* that part will be compiled and viewed.
15. 4allTEX contains a *huge* set of documentation, including TEX/LATEX/METAFONT documentation and tutorials both for novices and gurus, including:
    - *all* MAPS issues (both Dutch and English articles; starting in 1988; about 2000 pages with high density information arranged in a large set of PostScript files),
    - *all* TEXHAX issues (1986-1995),
    - *all* TEX-NL issues (1989-1995),
    - *all* TEX-MAG issues,
    - *all* UKTEX issues (1988-1995),
    - A lot of tutorials in TEX/LATEX source (also .DVI, .PDF, and .PS files available),
    - The 'TEXbook' and the 'METAFONTbook' in TEX source format,
    - FAQ about TEX, PostScript, etc,
    - WEB/literate program related contributions.

    All this documentation is available in PostScript as well as in PDF format. Using Ghostscript or the PDF reader (both on the CD-rom) you can print the documentation on any printer.
16. 4allTEX contains SPECIALS: chess (including chinese), bridge, music, crossword, go, and more; *including all fonts*.
17. 4allTEX contains Arabic, French, Cyrillic and Polish packages.
18. With 4TEX, it is very easy to generate completely new formats.
19. 4allTEX contains a lot of .DVI, and PostScript utilities.
20. 4allTEX contains GNUPLOT.
21. 4allTEX contains extensive bibliographies on TEX-related topics.
22. It contains a 4TEX '1.44 Mbyte 3.5" disc distribution version'.
    You can install only 'one disc' (a basic emTEX installation), *or* a number of discs, *or* all discs.

23. Because TEX is extremely useful for the generation of HTML, PDF and other multimedia documents you will find a lot of those programs on the CD-ROM. Also a lot of useful Internet programs are supplied.

The 4allTEX CD-ROM contains even more than that!!!

Hard disk requirement for running 4TEX (processing all your documents): less than 50kb!

The 4allTEX CD-ROM is the *first* and (at least in the near future) the *only* CD-ROM with a:

ready-to-use MS-DOS full TEX implementation

including a *huge* set of documentation (*and* it also works under LINUX, OS/2 and WINDOWS in a DOS box).

The 4allTEX CD-ROM contains a *unique set* of TEX and LATEX software for both MS-DOS and non MS-DOS users!

Unbelievable but TRUE!

3 Computer system requirements

The ready-to-use 4TEX system is available for PC MS-DOS 8086/80286/80386/80486/pentium systems. However, the *major* part of the software on the 4allTEX CD-ROM is system *in*dependent.
Further you need:

- a hard disk/diskette drive of 50kb (yes: 50.000 *bytes*), or more. . . ,
- a CD-ROM drive (of course),
- no memory requirements,
- any printer.

4 How to install

The 4TEX system on the 4allTEX CD-ROM can be installed within 1 minute (!) and the first TEX example can be previewed and printed within the next 10–20 seconds. After an installation of the CD-ROM i.e. after running the installation program `inst4tex` one can start 4TEXby typing

```
> 4tex sample
```

With the current installation program it is also possible to copy (parts) of the emTEX system onto a hard disk (in case you want to speed things up).

5 Price

- The price of the double CD-ROM including a 110+ pages manual (when ordering from the NTG office) is:

  **** 60 Dutch guilders ****

  (*or the equivalence in dollars*).
- The price includes shipping (even by air mail for outside Europe!)
- *Add 15 guilders* if you pay to NTG with a bank cheque or non-Dutch bank transfer (this is *not* recommended and costs the NTG/you the additional 15 guilders).

6 How to order the 4allTEX CD-ROM

There could be a possibility that **TEX Local User Groups (LUGs)** are also (re)selling the 4allTEX CD-ROM. *So please contact your local LUG first!*
NTG is selling the 4allTEX CD-ROM directly to all LUGs for bookstore pricing.

At the moment we are rearranging the way one can order the CD-ROM. Just wait until the official announcement or contact the NTG office at the end of May.

NTG
P.O. Box 394
1740 AJ Schagen, The Netherlands
e-mail: `ntg@nic.surfnet.nl`

7 4TEX discussion list support

Yes there is already a lot of 4TEX support using the world-wide 4TEX discussion list.

How to subscribe
send the one line message:

```
subscribe 4tex your_personal_name
```

to the listserv manager:

```
listserv@nic.surfnet.nl
```

How to use
send your questions/bugs/remarks/suggestions to:

```
4tex@nic.surfnet.nl
```

How to retrieve old e-mails
All discussions will be archived. In order to receive a list of all archived files, send the one-line message:

```
INDEX 4TEX
```

to the listserv manager:

```
listserv@nic.surfnet.nl
```

In order to retrieve one of the monthly archieved e-mail file, send the one-line message (for month 05 in year 94):

```
GET 4TEX LOG9405
```

to the listserv manager:

```
listserv@nic.surfnet.nl
```

The 4TEX authors Wietse Dol (`w.dol@lei.agro.nl`) and Erik Frambach (`e.h.m.frambach@eco.rug.nl`) are monitoring this fast growing 4TEX list!
For very special requests/questions use:

```
4tex-support@eco.rug.nl
```

4TEX discussion list e-mails can also be found on the 4allTEX CD-ROM.

8 The 4TEX manual

The price of the 4TEX manual only (without the 4allTEX CD-ROM) is:
In Holland/Belgium: f. 20,– (including shipping).
Other countries : $ 15 (including shipping).

9 4allTEX ftp availability

On anonymous ftp-site:

```
archive.cs.ruu.nl
```

in directory:

`pub/TEX/MSDOS/4alltex`

or on CTAN in directory:

`pub/tex/systems/msdos/4alltex`

you will find:

- the 4allTEX distribution version (an earlier version of 4TEX) in directories of about 1.44 Mbyte.
- updates of the 4allTEX CD-ROM.

10 Latest update

Just a quick list of the additions in 4TEX version 3.27 compared to the earlier release 3.25:

- 4TEX has extended the documentation with HTML files, PDF files etc.
- All documentation is available in PostScript format but also in PDF format. With the PDF reader it is possible to print the documentation on any printer/system.
- (as complete as possible) version of LaTeX2ε (patch level 3) (e.g. with PSNFSS),
- upgrades DVI-drivers from Eberhard Mattes (e.g. it is now possible to preview with the VESA resolution 1024x768),
- upgrades emTEX
- updates of the 4TEX batchfiles, in order to e.g.:
  - include your own PostScript fonts into TEX;
  - include your own PostScript family into TEX(by using the fontinst package);
  - printing to a PostScript file/printer using partial fontloading
  - the usual bugfixes (and introducing new ones)

11 Future improvements

Of course 4TEX is continuously under development. In a next release you can expect e.g.:

- latest version of other (TEX-related) products,
- update DVI drivers,
- and more.

If some essential packages and/or files are missing on the 4allTEX CD-ROM, please contact the authors on:

`4tex-support@eco.rug.nl`

Some of the above extensions and further improvements will be available as additional installation files from the internet archives and from the Dutch FGBBS bulletin board service of NTG (phone: +31 85 217041).

For more information, please contact:

Dutch (language oriented) TEX Users Group (NTG)
P.O. Box 394
1740 AJ Schagen
The Netherlands
e-mail NTG board: `ntg@nic.surfnet.nl`

Some Announcements from Usenet*

(edited by Gerard van Nes)

1 Partial Font Downloading utility for PostScript files[1]

Basil K. Malyshev
CERN European Lab for Particle Physics Sun, Geneve, Switzerland.
malyshev@dxcern.cern.ch
January 29th, 1995

I have uploaded into CTAN upgrade of the 'Partial Font Downloading utility'. It is in directory:
/tex-archive/fonts/utilities/fontload

This utility parses a PostScript file (by GhostScript program), determines fonts and characters which are used in it and loads required ATM compatible Type 1 PostScript fonts. Fonts are partially downloaded into file, i.e. only those characters which are really required for document printing.

This utility was tested with PostScript files writen by:
1. Rokicki's DVIPS;
2. Frame Products (under UNIX);
3. MS-Windows PostScript printer driver.

Job of this utility is similar to Partial font downloading facility of the DVIPSONE program which is distributed by Y&Y. However, DVIPSONE handles DVI files but not PostScript.

This utility automatically handles picture fonts and has no problems with mixing of the PostScript output from different systems. Of cource, this works slower than DVIPSONE because of PostScript file parsing.

This utility works in UNIX, VMS and IBM-PC/MSDOS. To operate it requires GhostScript version 2.6.1 (recommended) or GS 3.12.

The current version has following differences with previous version issued 28-Nov-94.
1. It generates more compact code to download font into printer, that reduces output file size and printer downloading time.
2. Printer memory requirements is also slightly reduced.
3. Ported to other platforms, such that it works now on UNIX, MS-DOS, VMS.
4. Improved control of the error situations.
5. Added option `-p` that selects font set which is built in printer.
6. Solved disadvantage with those fonts which have common CharStrings (for example Univers fonts).
7. Made the same font map for GhostScript and SubFont program.
8. Tested with GhostScript 3.12, that is able to apply partial font downloading to document used Multiple Master fonts. *However*, GS 3.12 has a bugs, for this reason GS 2.6.1 using is recommended.
9. Man pages for `fload` and `subfont` are added.

— — * — —

2 URL search for CTAN ftp://ftp.SHSU.edu/search.html

George D. Greenwade
bed_gdg@shsu.edu
March 8th, 1995

A README file containing some documentation for a new feature at the `ftp.shsu.edu` anonymous ftp archives (North American home of the CTAN)[2]. For inquiring minds, the URL for the 'search.html' file being referred to is

`ftp://ftp.shsu.edu/search.html`

I will be in contact with the other CTAN sites to see if they are interested in putting this on their nodes.

The file 'search.html' is a Hypertext Markup Language shell which is designed to provide the interactive functionality of the popular and widely-used anonymous ftp `quote site index` command (see `README.site-commands` and `README.archive-features`) to users of forms-capable World Wide Web browsers. Upon selecting this file with an appropriate browser, the user is prompted for a string to search for within the master index of files (INDEX in the top-level directory) archived in the Sam Houston State University anonymous ftp archives on `ftp.SHSU.edu`. This is not a WAIS-based or index-based search; instead, it is a more or less standard 'grep' search of that file. Thus, the search string provided may include grep-type special characters, if desired.

Two special distinctions are provided in `search.html` as compared to the anonymous ftp 'quote site index' command. First, rather than limiting the search results to the first 20 matches within the INDEX (a feature specifically

*In this contribution you will find some sofware announcements gathered by your editor. Do you find any interesting new or updated product on any discussion list? Please contact him!

[1] Available on the 4allTEX CD-ROM and implemented within 4TEX!

[2] CTAN = Comprehensive TEX Archive Network, sites which you can use anonymous ftp to obtain TEX/LATEX-related material from. CTAN is the 'home' of the official version of LATEX etc. CTAN sites are: `ftp.dante.de`, `ftp.tex.ac.uk` and `pip.shsu.edu`

designed into 'quote site index' for anonymous ftp to try and keep the search results on a single screen), all matches are returned to the user. Second, all file and directory names which are returned as matches to the request are automatically converted to HTML links to those files. Therefore, all matches within the archive can be found and immediately clicked on to retrieve the file or go to the directory desired. In addition to the active link, the usual 'quote site index' information — file size, archival date, and path-to-file — is included; the path information is simply 'hot'.

— — * — —

3 March 95 release of DECUS TEX (OpenVMS)

Christian Spieler
Technical University Darmstadt, FRG (Nuclear Physics Institute)
`spieler@linac.ikp.physik.th-darmstadt.de`
March 9th, 1995

Hello VMS TeX users!

I want to announce the new March95 release of DECUS TEX for OpenVMS.

The new release consists of some documentation files (names starting with "`0`...") and a bunch of ZIP archives. It can be found in the subdirectory tree starting with
`/tex-archive/systems/vms/`
on the CTAN archive sites and their mirrors.

DECUS TEX for OpenVMS provides a full featured TEX system on VAX and Alpha AXP computers running the current OpenVMS 6.1. The previous OS releases (VMS 5.5 on VAX and OpenVMS 1.5 on AXP) should work, too.

Some highlights of the new release:

- Now, all programs of the Knuth core distribution are supported, even the 'obsolete' `.PXL` file utilities:
  - added `patgen`
  - added `mft`
  - added `pktogf, gftopxl, pktopx, pxtopk, vftovp, vptovf`
- Bugfixes in
  - tangle (important, use this new release to recompile other TEX/METAFONTsources)
  - vfware programs
  - TEX and METAFONT command line interface
- Updated DVIPS to version 5.58
- Now, METAFONT is BigMETAFONT(262142 words main memory)
- All 'core programs' (Web, TEX, METAFONT, BibTEX, TEX/METAFONT/VFware) support wildcards in input file name specifications. This allows: *automatic single/multilevel subdirectory searching!!!*
- In the 'large' programs (TEX, METAFONT, BibTEX) that use the VMS CLI system utility for command line parsing, a default command table is now included in the executable. This allows installing these programs as 'foreign commands'.
- The command line interface of the utility programs has been enhanced:
  - They use more useful defaults when (parts of the) arguments are missing.
  - The command line supports multiple arguments for those programs where this makes sence, to allow specification of input and output files on the command line.

For more information, please see the `0*` files.

— — * — —

4 Release version of PSfrag for LATEX2e

Michael C. Grant
Information Systems Laboratory, Stanford University
`t@isl.stanford.edu`
`mcgrant@rascals.stanford.edu`
March 16th, 1995

The release version of PSfrag for LATEX2ε can be found at CTAN sites in the directory: `macros/latex/contrib/supported/psfrag`

4.1 What is PSfrag?

Many drawing and graphics packages produce output in PostScript, but do not support the inclusion of equations and other scientific text of which LATEX is capable. Likewise, many LATEX users simply find the various LATEX graphics packages too clumsy, and prefer the familiar GUI of a PostScript-generating graphics tool.

PSfrag provides the best of both worlds, by allowing the user to replace arbitrary text in Encapsulated PostScript files with arbitrary LATEX constructions. One can place a simple text 'anchor' in the graphics file to denote the position of a desired LATEX equation (for example), and PSfrag will automatically remove that anchor and replace it with the properly sized, aligned, and rotated LATEX equation.

The full documentation for PSfrag found in the release contains examples and usage instructions.

4.2 To use PSfrag

You will need:

- LATEX2ε and the 'graphics' package.
- Any dvips driver which supports the `\special{ps: }` command for inserting raw PostScript code. The package has been tested only on the 'dvips' PostScript driver (courtesy Tom Rokicki).
  If you have another driver, and can patch PSfrag to work with it, please let me have the patch! I'll try to work it into the code. This should not be difficult if your driver works well with graphics.sty; in fact, in the future, all compatibility issues will be determined solely in `graphics.sty`.
- Perl 4.036 or later.
- The Ghostscript PostScript interpreter.

These last two programs are only necessary if you wish to process new figures for use by PSfrag. If you only wish to run LaTeX on documents whose figures have *already* been processed, then Perl and GhostScript are not necessary.

4.3 (Important) changes from LaTeX2.09 PSfrag

This is a major release of PSfrag, and its operation has changed as much as LaTeX itself has of late.

- As past users of PSfrag are aware, the PSfrag system must first process each Encapsulated PostScript file to determine the position and rotation of each piece of text in the figure. In LaTeX2$_\varepsilon$, this information is stored *in the original PostScript file* as a set of additional Document Structuring Comments. Before PSfrag 2.0, this information was stored in a separate file. These comments have NO effect on the original PostScript file. (PSfrag 2.0 does, however, have the ability to read old-style PSfrag info files, for back-compatibility purposes.)
- A new alignment mode, 'B' (baseline), has been added. This causes the alignment point to occur at the baseline of the text instead of at the true bottom of the bounding box. The default alignment has been changed from 'bl' to 'Bl'.
- Since LaTeX2$_\varepsilon$ does not support dvips' `epsf.sty` macros, (although they do happen to work), PSfrag 2.0 has been designed around the 'graphics' package instead. Therefore, you must either include the PostScript figures with `\includegraphics`, or use the `epsf.sty` wrapper provided with PSfrag 2.0 (or any other wrapper, for that matter). The `epsf.sty` package should provide seamless back-compatibility with your old documents, too.
- The scaling and resizing operators in `graphics.sty` will scale the PSfrag text as well. To prevent this from happening (i.e., to maintain the old behavior), use the key-value sizing commands of `graphicx.sty`, or use the LaTeX2$_\varepsilon$-ified `epsf.sty` or `epsfig.sty`.

4.4 Changes from the beta release

PSfrag 2.0 has undergone beta testing, and as a result a few changes have been made:

1. Numerous bug fixes.
2. Support for both methods of naming LaTeX 2.09-style `*frag` files is supported: `example.eps` $\longrightarrow$ `example.epsfrag`, and `example.eps` $\longrightarrow$ `example.frag`. If you are running on UNIX, it will always look for both (in the above order). Of course, this is irrelevant in LaTeX2$_\varepsilon$ mode.
3. The ps2frag script now handles both LaTeX2.09 and LaTeX2$_\varepsilon$ modes. In other words, it will generate '*frag' files for your figures if you specify the '-209' option; so it can completely replace the LaTeX-209 script.
4. The package option '209files' has been changed to '209mode'.
5. A complementary option, '2emode', has been provided when LaTeX2$_\varepsilon$ is run in 2.09 compatibility mode, which allows documents which must remain in 2.09 style for other reasons to access the full functionality of LaTeX2$_\varepsilon$'s PSfrag (the resulting documents may only be processed by LaTeX2$_\varepsilon$, however).
6. `ps2frag.ps` has been included in `psfrag.dtx`, so it can be documented more completely.
7. the documentation has been broken out into a separate file. psfrag.dtx still contains the commented code, but the users guide is now in pfgguide.tex,ps.

4.5 Bugs? What Bugs?

PSfrag 2.0 has undergone a beta test, and I have personally used it on a significant number of figures. Its backwards compatibility mode has undergone testing as well. This is not a commercial package, so you can expect it to be bug-free. Just kidding!

If you have problems, bug reports, or improvement suggestions, please send them to the PSfrag maintainer's mailing list,

`psfrag@rascals.stanford.edu`

You may be asked to supply examples to demonstrate the behavior you wish to have corrected or improved.

This is a majordomo mailing list; feel free to join if you would like to actively participate in PSfrag development. It is not a general user's list; if demand is high, however, one can be created.

Thanks for using PSfrag!

— — * — —

5 LaTeX package for writing chemical symbols

Mats Dahlgren
Lab. for Chemical Surface Science
The Royal Institute of Technology
S-100 44 Stockholm, Sweden
`matsd@physchem.kth.se (mats dahlgren)`
Ph: +46-8-790 8596; Fx: +46-8-790 8207
26 Apr 1995

Now there is a package for LaTeX2$_\varepsilon$ avaliable at CTAN which simplifies in writing chemical texts. The package's name is 'chemsym', from 'CHEMical SYMbols'.

The package defines commands for the chemical symbols of the elements, which always are type-set in upright font, as they should. If not followed by a subscript, superscript, '(', or ')', a small space is inserted. The package also makes the TeX/LaTeX commands '^' and '_' possible to use in text mode, to facilitate typing chemical formulas and superscripts to units.

The package can be obtained from CTAN (*) in the directory

`/tex-archive/macros/latex/contrib/supported/chemsym`

Announcement from the LaTeX3 Project team

Frank Mittelbach
Technical Director

Mainz, December 6, 1994

We are pleased to present the final report on 'Math Font Encoding' produced by Justin Ziegler for the LaTeX3 project. This document is electronically available from the CTAN hosts in the directory

```
/tex-archive/info/ltx3pub
```

To process the document you will need the files

```
l3d007.tex
l3ms002.cls
```

The document is about 90 pages long. In one of the appendices, there are three font tables using fonts which are often not part of a LaTeX installation. However, you can process the rest of the document successfully without them (just ignore the error messages they generate:-).

The rest of this announcment is lifted straight from the preface of the document.

Justin has worked for three months at the Johannes Gutenberg University Mainz. His work was generously sponsored by GUTenberg (The French TeX Users Group) and by the ZDV of the University of Mainz (Data Processing Center), the latter providing Justin with office space and taking care of the administrative details.

In the past years a lot of work went into integrating new fonts into the TeX system. Only five years ago, typesetting with TeX basically meant typesetting in Computer Modern. Nowadays many users can choose (at least theoretically) from several thousands of fonts. Today, NFSS is the standard font selection in LaTeX and due to this mechanism and the fontinst-package by Alan Jeffrey virtually every PostScript font, in fact, every font for which a `tfm`-file can be obtained, can be used, out of the box, with LaTeX.

But for these thousand text fonts there are only five font families for use in math formulas to go with them. Even worse, every of these math font sets are encoded in a different way making it nearly impossible even for an expert TeX user to use different fonts for math in different jobs.

The work undertaken by Justin is the first of several steps to solve the problems at hand, the final goal being the development of a system that allows the user to change math fonts as painlessly as it is now possible with text fonts.

Based on Justin's analysis and his proposal, the LaTeX3 Project is now undertaking to provide a prototype implementation for math fonts, starting with the Computer Modern fonts as well as the Euler Math fonts. We expect this implementation to be available for public usage during 1995-96.

Diplomatic edition of a medieval Icelandic manuscript*

The making of a scholarly edition

Andrea de Leeuw van Weenen

Rijks Universiteit Leiden, Vakgroep VTW
Postbus 9515
2300 RA Leiden
lettvarulmvs.LeidenUniv.nl

In november 1993 my edition of the Icelandic Homily Book was published by the Stofnun Árna Magnússonar á Íslandi[1] after having been 'in print' for a period of 19 years. If it had not been for TEX, this period might easily have been extended indefinitely. Looking back, work on the Icelandic Homily Book can be divided into three stages: the scholarly work, the attempts at printing before TEX, and the typesetting with TEX.

1 The scholarly work

My involvement with this edition, or with Old Icelandic scholarship in general, came by almost by accident. I arrived in Iceland in 1971 with my husband, who had taken a temporary job at the University of Reykjavík, and my two small sons, and my knowledge of Icelandic at that time could easily find place in half a column of the MAPS. In order to escape the drudgery of diaper laundry I enrolled in the *Icelandic for foreigners* program at the university where I got enthralled in my second year by the secrets of paleography and Old Icelandic grammar. So when I had passed my examination, I looked around for something useful in that direction to occupy me in my third and final year in Iceland. The suggestion by Helgi Guðmundsson, associate professor of Icelandic at the University of Reykjavík, to write a doctoral thesis and to choose an edition with a thorough grammatical analysis as the topic did not strike me as a realistic option. I had majored in mathematics, so would have to go a long way before getting to a doctorate in a completely different field. However, he insisted that shortcuts could be found and that doing the edition while I had the right resources was a sensible thing. Although I did not believe him at the time, he turned out to be right. Anyway, I let myself be talked into this undertaking and after some consultations with the Stofnun Árna Magnússonar I choose the Icelandic Homily Book from the three or four manuscripts that the institute and Helgi deemed suitable and most urgent. Icelandic Homily Book is apart from some fragments the oldest extant Old Icelandic manuscript, dating from around 1200 and containing on its 102 parchment leaves (204 pages) some 60 sermons. This manuscript is by its age alone of the greatest interest for the study of the Old Icelandic language, but it is also considered to be an example of good style.

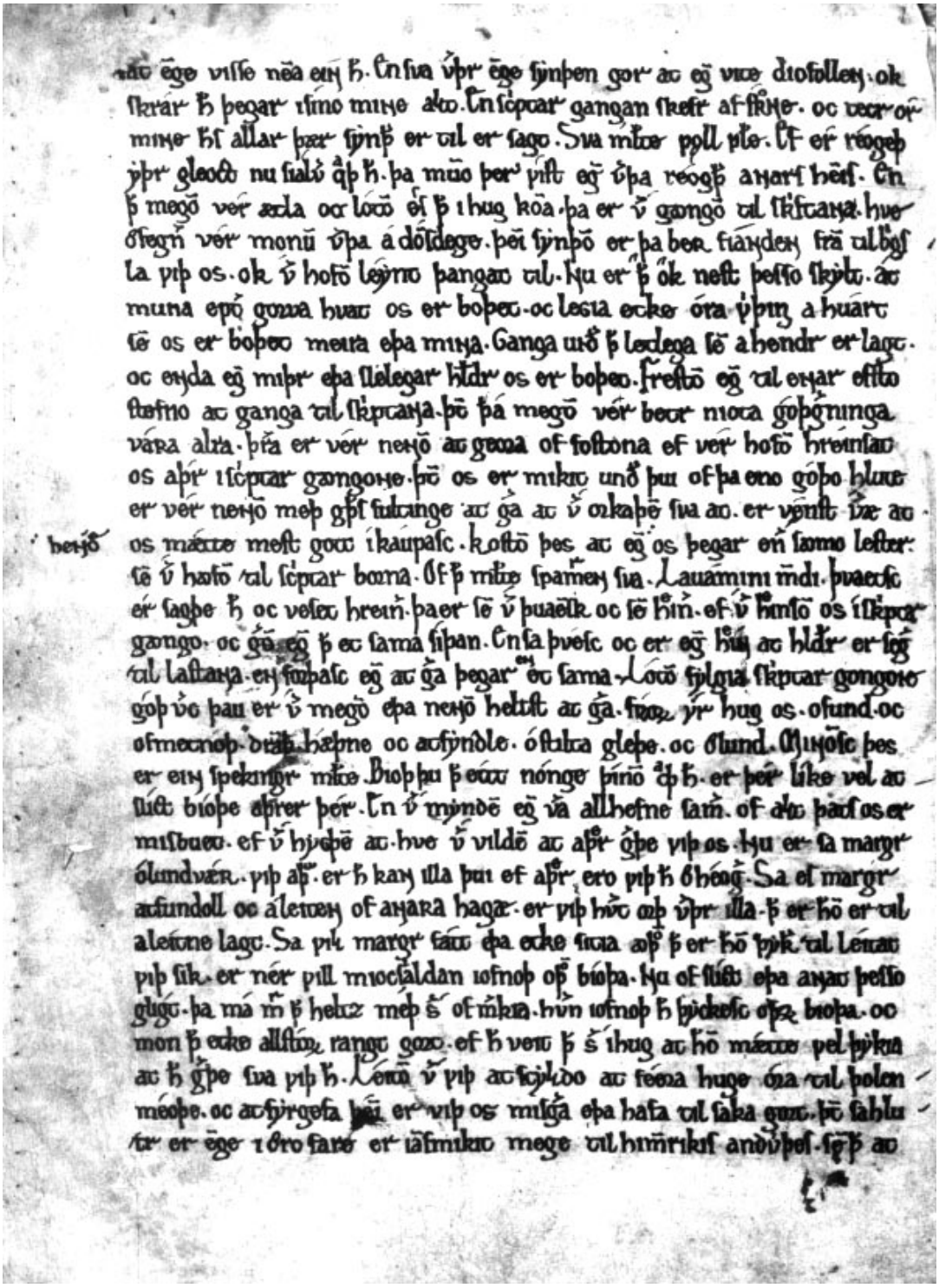
Work on the transcription started in the summer of 1973. After the first year the transcription with the critical apparatus was finished, and the introduction which was going to concentrate on orthography and morphology was well under way. Meanwhile, the staff of the Stofnun Árna Magnússonar had been keeping an eye on my work, and had offered to publish the edition in one of their series as a combined facsimile and diplomatic edition. I gladly accepted their offer, but should perhaps have been forewarned for the problems encountered afterwards when the meeting devoted to this project was nearly exclusively devoted to the choice of paper, instead of to editorial principles, deadlines to be met, special requirements and the like. So I left Iceland in 1974 with the promise that typesetting the transcription would start next week. Famous last words. During the next two years I finished writing the introduction and fulfilled the requirements of the University of Utrecht for a MA in Old Germanics. As typesetting in Iceland still had not started, I typed the introduction, pasted the needed corrections into the transcription and handed the thesis in as typescript, thinking that it well might be some more years before the book got printed, but never suspecting that it would take 17 more years, or that I would have to be my own typesetter.

2 Typesetting, the years before TEX

In 1974 all typesetting on Iceland was still done in lead. The transcription required a number of unusual characters and it turned out that not only did the typesetting firms not have these characters, they did not exist in the Monotype catalogue. So they would have to be specially cut for this edition. The various firms that were approached were understandably reluctant to invest in this, as there was no guarantee that the characters could be used for other books. These negotiations took several years as in the small Icelandic community firms could be approached only one at a time and most took their time to think the proposition over. In 1979 the news came that one firm had purchased phototypesetting machinery of the matrix variety and that they

*Presented at the 14^e NTG meeting November 17th, 1994, Antwerpen, Belgium.

[1] This institute in Reykjavík keeps most of the existing Icelandic manuscripts and is devoted to their study and publication.



were willing to start work on the transcription. Slowly, the proofs started to come. But with them came a surprise. I had believed that proofreading would be my responsibility, but now I found that proofs of books to be published by the Stofnun Árna Magnússonar were habitually read by three independent readers, the editor of the edition, one of the senior staff members and a junior staff member, and that proofreading not meant comparing the proofs with the typescript, but with the manuscript or the photographs, thus checking not only the work of the typesetter but also that of the editor. This meant that proofreading took quite some time. For the staff of Stofnun Árna Magnússonar it was one of the many jobs they had to do besides their own research. And when we disagreed about a reading there were lengthy discussions by mail, which usually got only solved during one of my visits to Iceland. So when we finally were in agreement about the corrections to be made and sent the corrected proofs to the typesetter, it was a very unpleasant surprise when we were told that he had just got himself a new phototypesetting machine and could not convert the

at enge viſſe nema éiN haN. En ſva verþr enge ſynþen gor at eige vite diofollen. ok
ſkrár haN þegar iſíno miNe aLt. En ſcriptar gangan ſkefr af fkróNe. oc tecr ór
3 miNe hanſ allar þær ſynþer er til er ſagt. Sva mælte poll poſtole. Ef ér réogeþ *I Cor 11.31*
yþr gleoct nu ſiálver quaþ haN. þa muno þer víſt eige verþa réogþer anarſ heimſ. En
107 þat megom vér ætla oc lótom | ø\ſ/ſ þat i hug koma. þa er vér gængom til ſkriftana. hve
6 ôfegner vér monum verþa á dómſdege. þeim ſynþom er þa ber fiándeN fram til brigſ-
la viþ os. ok vér hofom léynt þangat til. Nu er ok þat neſt þeſſo ſkylt. at
muna epter gorva hvat os er boþet. oc legia etke óra virþing a huárt
9 ſem os er boþet meira eþa miNa. Ganga under þat letlega ſem a hendr er lagt.
oc enda eige miþr eþa ſlélegar heldr os er boþet. Freſtom eige til enar efſto
ſtefno at ganga til ſkriptana. þuiat þá megom vér betr niota góþgerninga
12 vára alra. þeirra er vér nenom at geora of foſtona ef vér hofom hréinſat
os áþr i ſcriptar gængone. þuiat os er mikit under þui of þa eno góþo hlute
er vér nenom meþ guþſ fultinge at gera at vér orkaþem ſva at. er vęnſt være at
15 os mætte meſt gott í káupaſc. Koſtom þes at eige ´hende` os þegar ener ſæmo leſter.
´ ſem vér hæfom til ſcriptar borna. Of þat mælte ſpamaþren ſva. Lauámini mundi. Þvaetſc *Is 1.16*
ér ſagþe haN oc veſet hreiner. Þa er ſem vér þuaemſk oc ſem hreiner. ef vér hreinſom os íſkriptar
18 gængo. oc gerom eige þat et ſama ſíþan. En ſa þvéſc oc er eige hréin at heldr er ſeger
til laſtana. en forþaſc eige at gera þegar `en´ et ſama. Lótom ſylgia ſkriptar gongone
góþ verc þau er vér megom eþa nenom heltſt at gera. ſnorum ýr hug os. ofund. oc
21 ofmetnóþ. dramb. hæþne oc atfyndle. ó ſtilta gleþe. oc ôlund. Minomſc þes
er ein ſpekingr mælte. Bioþ þu þat éitt nónge þínom quaþ haN. er þér líke vel at -
ſlíct bióþe aþrer þér. En vér myndem eige vera allhefne ſamer. of aLt þatſ os er
24 miſbuet. ef vér hygþem at. hve vér vildem at aþrer gerþe viþ os. Nu er ſa margr
ólundvæR. viþ aþra. er haN kan illa þui ef aþrer ero viþ haN ô héoger. Sa eſ margr
atfundoll oc á leiten of anara hagæ. er viþ hvert orþ verþr illa. þat er honom er til
27 a léitne lagt. Sa viL margr fátt eþa etke ſitia æþrom þat er honom þyker til léitat
viþ ſik. er nér vill mioc ſ\i/aldan iofnoþ oþrom bióþa. Nu of ſlíct eþa anat þeſſo
glígt. þa má maþr þat heltz meþ sér of merkia. hvern iofnoþ haN þyckeſc oþrum bioþa. oc
30 mon þat etke allſtórum rangt gort. ef haN veit þat sér ihug at honom mætte vel þykia
at haN gerþe ſva viþ haN. Léitom vér viþ at ſcyLdo at féora huge óra til þolen-
méoþe. oc at fyrgeſa þéim er viþ os miſgera eþa haſa til ſaka gort. þuiat ſa hlu-
33 tr er enge i ôro fare er iamſmikit mege til himenrikiſ andvirþeſ. ſem þat at

2 fkróNe] ɔ: ſkróNe 7 ok þat] *T*. 10 heldr] *Add* en (*Vr*). 11 þá] < þat. 20 ſnorum] n < v, *correction indicated by an accent*. 21 ofmetnóþ] *Accent suspect, L* ofmetnoþ. 26 hagæ] æ < a. 28 nér] *Del* (*SK*). 31 féora] o < e.

material he had on punch tapes to this new machine. But he would have the thing typeset anew as soon as he could.

And so the whole circus started again: Proofreading in triplicate. There were less cases to discuss between us three, but on the other hand the work went a lot slower. I was both in the final stage of another project and taking up a new job which required a lot of reading up, and if there had ever been any feeling of urgency about the book in Iceland that had now certainly gone. So it was early 1989 when the marked proofs were returned for the second time to the typesetter. But when I arrived in Reykjavík some months later, I found that the machinery had again been replaced and that the typesetter was planning to start from scratch again. At that time I had about 10 years experience with computers and I was quite sure that conversion was possible. Moreover, I had at some stage requested and got copies from the typesetting files. Admittedly it had not been easy to decipher those, but I had copies on DOS disks of the original files and conversions of these files to ASCII where the typesetting codes had been removed. At this stage the Stofnun Árna Magnússonar was as opposed as I was myself to going through the whole troublesome procedure again as it was getting clear to us that with the methods of the institute we would always be limping behind the pace of technology.

So disks were sent to Iceland and in due time new proofs arrived. But after the initial joy that conversion to the new machine turned out to be possible, a closer look brought great disappointment. The font used looked decidedly irregular and the kerning of the high s (ſ) was absolutely ugly, but worse, lots of errors had crept in. A systematic study of the errors I found on the first few pages brought me to the conclusion, which was later affirmed, that a conversion program had been written, but that when this was found not to produce the correct result, rather than correcting and rerunning the program the output file had been corrected, and that not very systematically. From this level of competence to judge I decided that the safest way would be to get my hands on their files and to repair those by comparison with mine. As this required only a physical conversion to DOS disks it seemed not to tall an order. However, this could not be done in Iceland, but had to be handled in Denmark, and after some phoning and explaining 2 disks arrived, which were not too difficult to decipher. As soon as I had corrected a couple of pages I returned a disk, and waited with some optimism for a corrected proof. No such thing, but a panicky fax that the disk could not be read. Some weeks of multilateral discussion followed between the institute and the typesetter in Reykjavík, the technical staff of the manufacturer of the typesetting machine in Denmark, and myself in Leiden. This discussion was not made any easier by the lack of a common language. In the end it became clear that the lack of expertise on the Icelandic end combined with the distances involved made it highly unlikely that the problem would ever be solved.

At that time I had some experience with TEX, enough at least to be confident that the job could be done, and luckily not enough to foresee all the problems involved. So I wrote a letter to Iceland enumerating the possibilities open to us, from starting from scratch with typesetting for the third time via various methods involving conversion to doing it myself with TEX, stating the adhering disadvantages and advantages and the fact that in my opinion some methods were so impractical and relied so much on factors without our influence that I was not willing to cooperate in them. Probably the members of the staff of the Stofnun Árna Magnússonar were then about as fed up with the whole thing as I was, so they agreed that I should have a go with TEX.

3 Typesetting, the years with TEX

As the book had to appear in a series and was planned as a combined facsimile and diplomatic edition with photographs and transcription on facing pages both page breaks and line breaks were decided by the manuscript, not by the software. The large paper size required a 12 point font. It came therefore as an unpleasant surprise that the cm fonts which I wanted to use were significantly wider than the fonts used previously and, more to the point, that the resulting lines did not fit the given page width. After much hesitation I decided to decrease the width of the characters about 10%.

The paperweight too was not unproblematic. Some manuscript pages had far more lines than others and there was a critical apparatus too that had to be accommodated as a whole at the foot of the page. If I choose a page height that would fit all pages, the majority would look ugly. So after some experiments I choose a page length that fitted most pages with the apparatus at the bottom of the page. The overlong pages had a special page height and the apparatus directly following the text.

The next problem were the special characters that had caused us problems right from the beginning: ſ aꝛ ǫ aꝏ to name a few. Some were easily made like the high s which just required removing the bar from f, and of course the introduction of quite a few new ligatures. Others required more METAFONT skills, like aꝛ or ⅁.

The transcription has small capitals within words otherwise consisting of romans. Normally small caps are larger than the corresponding romans. This made the page look very jumpy, so I scaled down the small caps. This was not completely successful. I feel that a small cap that has to fit within a word should be parameterized in a different way, but for that task I lacked the time.

The transcription also has italics and romans mixed within words. I had thought that the italic correction would take care of that problem, but it did not. So I had to figure out experimentally the amount of kerning needed for each pair of roman-italic and italic-roman that occurred. Again, this can certainly be improved upon by someone with a designer's eye. I can only say that this kerning is a great improvement upon the results without the kerning. The TEX files for the transcription pages were produced by program from the original ASCII files, so the program could insert explicit kernings as well. However, the introduction still only existed as a typescript and contained thousands of quoted words from the transcription. I was not looking forward to typing in that amount of explicit kernings, so I decided to solve the kerning problem by combining romans and italics in a single font and take care of the kernings in the ligature tables. The small caps and the italic small caps which occurred within the transcription were placed in this same font, and a lot of macro's defined. As mostly only one or two consecutive italic characters occur, this made typing not to strenuous.

Apart from the adaptation to the width of the characters the cm part of this combined font had undergone only one change: ø. The height of this character is the height not of the o, but of the diagonal stroke. This results in the accent over ø standing higher than that over o: ǿ ó. By reducing the height of the ø to the height of o the accents come at the same height: ǿ ó.

Het digitaal produceren van een proefschrift

Ed Boets

Academisch Ziekenhuis Leiden, Afd. Oogheelkunde

maart 1994

Abstract

Het digitaal produceren van een proefschrift beschrijft hoe een proefschrift, in LaTeX gemaakt, als postscript-file via het modem naar de drukker gestuurd kan worden om vervolgens op een Docutech systeem (600 dpi) gedrukt te worden.

Keywords: proefschrift, docutech, digitaal, 4allTeX

1 Aanzet

In 1991, ik was nog in militaire dienst, kwam ik in contact met LaTeX, de persoon die het mij liet zien werkte destijds op een ATARI en hoewel het werken met de shell wel aardig was, vond ik de snelheid van compileren ('TeXen') te traag. Ik ging toen op zoek naar TeX voor de PC. De commerciële bronnen waren snel genoeg gevonden, maar ook prijzig. Via een kennis bij de Rijks Universiteit Leiden kreeg ik emTeX op 10 diskettes (inclusief HP fonts). Ondertussen was de militaire dienst afgelopen en was ik begonnen aan een promotie onderzoek bij de afdeling Oogheelkunde van het Academisch Ziekenhuis Leiden, waar ze niet verder waren dan 'Word Perfect' versie 4.2. Nu was versie 5.0 snel genoeg gekocht maar TeX en LaTeX kennende was ik net als vele NTG-leden niet tevreden over een heleboel kenmerken en vooral tekortkomingen van WP. Hoewel ik mijn artikelen voor wetenschappelijke publicatie nog wel in WP schreef (de harde schijf op het werk was met 20 Mb te klein om emTeX te plaatsen), ben ik me gaan verdiepen in de mogelijkheden om met LaTeX mijn proefschrift te schrijven.

2 Tekeningen

Al snel ontdekte ik dat het verwerken van grafieken en figuren in de tekst flinke problemen opleverde. In eerste instantie probeerde ik alle figuren die ik had om te zetten met 'BitMap2Font' maar het duurde enige tijd voor ik doorhad waar alle 'pk' en 'tfm' files geplaatst moesten worden. Met een simpel batch-bestand werd daarna alles automatisch gedaan. Het was in die periode van experimenteren met 'BitMap2Font' dat 4allTeX zijn (of haar?) intrede deed. Via Wietse Dol was ik altijd verzekerd van de meest recente versie. Wietse gaf mij ook de tip om eens met Erik Frambach te praten over zijn 'thesis' style. Via Internet waren de contacten snel gelegd en had ik in korte tijd het bestand 'Edthesis.sty' op mijn harde schijf staan. Daar is vervolgens het een en ander aan veranderd om het aan mijn wensen te laten voldoen. 'Figures.sty' was ondertussen ook gevonden zodat het omzetten met 'BitMap2Font' niet langer nodig was.

De tekeningen voor het proefschrift, zo'n 41 in totaal, waren afkomstig van verschillende tekenprogramma's die veelal niet zo'n mooie postscript uitvoer hadden (toch vaak bitmapped). Ik had de postscript tekeningen nodig aangezien ik het proefschrift op een hoge resolutie postscript systeem (Docutech, 600 dpi) wilde uitdraaien. Gelukkig hadden de meeste programma's (zoals Quattro Pro) wel de mogelijkheid om de tekeningen als HPG Plotter file uit te voeren, waarna 4allTeX daar keurig een EPS en een PCX plaatje van maakte. Dit heeft als voordeel dat als ik tussentijds een uitdraai van het proefschrift wilde maken op mijn eigen printer (Canon LBP4-plus met HP emulatie), ik toch alle tekeningen kon printen, gebruikmakend van de PCX tekeningen. De tekeningen van CorelDraw (versie 3.0) konden uitgevoerd worden als EPS, in de niet-bitmap versie, om vervolgens omgevormd te worden tot een PCX-formaat.

3 LaTeX-problemen

LaTeX-problemen waren er dankzij de 4allTeX CD-ROM nauwelijks. Een van de eerste proefdrukken (slechts enkele pagina's groot) liet al meteen zien dat de formules het grootste probleem waren. De tekst was namelijk in Times gesteld en de formules nog in Computer Modern. De postscript uitvoer gaf echter een zeer iele Computer Modern te zien. Gelukkig was de redactie van de Maps dit euvel ook al tegengekomen en hadden ze dit probleem opgelost met een aparte driver voor de Docutech. Via Internet en Erik Frambach had ik al snel de Docutech driver op mijn harde schijf staan. Op de nieuwe 4allTeX CD-ROM staat deze driver er trouwens standaard op.

4 Kosten

Uiteindelijk was het dan zover, het proefschrift was geschreven, alle tekeningen omgezet en in het proefschrift geplaatst. Nu kon ik offertes opvragen. De vier bedrijven die ik aangeschreven had voor een offerte (alle werkten

met het Docutech systeem) gaven zeer uiteenlopende prijsopgaven voor het binnenwerk. Voor de 400 exemplaren van 164 pagina's liep dat van *f* 4050,– bij 'Copy Systems' in Groningen en bij 'Repro Bussum' te Bussum, via ± *f* 5700,– bij 'De Printer' te Rijswijk tot zelfs *f* 8775,– (binnenwerk en 'wire-o binden') bij 'Holland Ridderkerk' te Ridderkerk. Het digitaal aanleveren is daarmee aanmerkelijk duurder dan het camera-ready aanleveren van een proefschrift.

Bij twee bedrijven was het mogelijk om het bestand (postscript formaat, 6.5 Mb groot, ARJ-gecomprimeerd 1.6 Mb) via een modem aan te leveren (Copy Systems en Holland Ridderkerk). Beide bedrijven hebben een Bulletin Board, Holland Ridderkerk zelfs met een gratis 06-nummer voor snelle modems. Via een 14.400 modem kon het bestand in ± 80 minuten verstuurd worden.

5 Conclusie

Het digitaal aanleveren van het proefschrift bleek duurder dan de oude manier van aanleveren van een 'camera-ready proefschrift'. Dit terwijl ik had gedacht dat door de verminderde hoeveelheid werk dat de drukker aan het proefschrift zou hebben, de kosten lager zouden zijn. Waarschijnlijk zijn de nog hoge afschrijfkosten van de nieuwe apparatuur en de moordende concurentieslag op het fotozetwerk terrein daar debet aan.

De keus viel uiteindelijk op Copy System in Groningen waar de kosten voor de omslag die aangeleverd werd als een CorelDraw bestand, beduidend lager waren dan bij Repro Bussum. Bovendien werden de stellingen (PostScript) en de uitnodigingen (in kleur via een CorelDraw tekening) gratis gedrukt en werden er 400 enveloppen bijgeleverd om de uitnodigingen te versturen.

Metafont's mode_def in action

Erik Frambach

Rijksuniversiteit Groningen, Vakgroep Econometrie,
Postbus 800, 9700AV Groningen
e.h.m.frambach@eco.rug.nl

Abstract

In order to obtain maximum output quality when using METAFONT for rendering bitmapped fonts you need to specify the characteristics of the intended output device. This is done by defining a mode_def in which several variables are assigned. The meaning and effect of these variables are discussed in a case study of two types of laser printers.

Keywords: METAFONT, mode_def, bitmapped fonts.

1 Introduction

Being familiar with Hewlett-Packard's Laserjet family of laser printers, I was much surprised when I saw output from a Xerox Docutech printer the first time. The latter is a 600 dpi PostScript compatible laser printer, and as such comparable to the HP Laserjet 4M.

Using resident fonts such as Times Roman the differences are subtle though obvious. But when using e.g. Computer Modern fonts generated for the Laserjet 4M the differences are dramatic. Lines are much too thin or even seem to disappear at small font sizes.

Obviously, fonts for the Docutech system need to be generated with different METAFONT parameters. And indeed, after generating new fonts specifically for the Docutech the output looked much better, even better than the Laserjet's output. METAFONT did its job very well, using a mode_def specific for the Docutech printer.

2 What is a mode_def?

A mode_def is a definition of a series of assignments to various device-specific variables. It tells METAFONT how to compensate for certain device-specific characteristics. They are usually stored in a file called local.mf that is typically used when making METAFONT base file (iniMF). The file local.mf may contain many mode_defs, though you may want or need to restrict it to the locally used printer types to avoid exceeding METAFONT's capacity.

Now that we know that mode_defs can make the difference between good and bad output we will examine the variables and compare the mode_defs for the Laserjet and the Docutech.

```
mode_def ljIV =
  proofing:=0;
  fontmaking:=1;
  tracingtitles:=0;
  pixels_per_inch:=600;
  blacker:=0;
  fillin:=.2;
  o_correction:=.6;
  enddef;

mode_def docutech =
  proofing:=0;
  fontmaking:=1;
  tracingtitles:=0;
  pixels_per_inch:=600;
  blacker:=1;
  fillin:=.1;
  o_correction:=0.9;
  enddef;
```

The variables in these definitions are explained in the METAFONT book, but Karl Berry's explanation in his mode_def collection[1] is quite sufficient for non-experts (page numbers refer to the METAFONT book):

aspect_ratio: the ratio of the vertical resolution to the horizontal resolution (page 94).

blacker: a correction added to the width of stems and similar features, to account for devices which would otherwise make them too light (page 93). (Write-white devices are best handled by a more sophisticated method than merely adding to blacker, as explained above.)

fillin: a correction factor for diagonals and other features which would otherwise be 'filled in' (page 94). An ideal device would have fillin=0 (page 94). Negative values for fillin typically have either gross effects or none at all, and should be avoided.

fontmaking: if nonzero at the end of the job, METAFONT writes a TFM file (page 315).

o_correction: a correction factor for the 'overshoot' of curves beyond the baseline or x-height. High resolution curves look better with overshoot, so such devices should have o_correction=1; but at low resolutions, the overshoot appears to simply be a distortion (page 93). Here

[1] Available at ftp.umb.edu: pub/tex/modes.mf or at CTAN: fonts/modes/modes-2.1.mf.

some additional comments about `o_correction`, courtesy of Pierre MacKay (edited by Karl):

At present, I find that `o_correction` works nicely at 80 pixels per em, and gets increasingly disturbing as you move down towards 50 pixels per em. Below that I do not think it ought to happen at all.

The problem, of course, is that full `o_correction` is supposed to occur only at the zenith and nadir of the curve of 'o', which is a small region at high resolution, but may be a long line of horizontal pixels at medium resolution. The full `o_correction` does not change a 300 dpi cmr10, but it changes a 21-pixel high cmr12 to be 23 pixels high. The extra height and depth is seen along a line of seven pixels at the bottom and five at the top. This is a pronounced overshoot indeed.

For high-resolution devices, such as phototypesetters, the values for `blacker`, `fillin`, and `o_correction` don't matter all that much, so long as the values are within their normal ranges: between 0 and 1, with the values approaching 0, 0, and 1 respectively.

`pixels_per_inch`: the horizontal resolution; the METAFONT primitive `hppp` (which is what determines the extension on the GF filename, as among other things) is computed from this (page 94). (An 'inch' is 72.27 pt in the TEX world.)

To be more precise, you can determine the resolution of a font given a `mode_def` and a magnification `m` by simply multiplying `pixels_per_inch` for that `mode_def` by `m`. (Your results may differ from METAFONT's if you don't use equivalent fixed-point arithmetic.) Then you can determine the number used in the name of the GF font output by rounding. For example, a font generated at `magstep(.5)` (which is $\sqrt{1.2}$, which METAFONTcomputes as 1.09544) for a printer with `pixels_per_inch=300` will have a resolution of 328.63312 dots per inch, and the GF filename will include the number `329`.

`proofing`: says whether to put additional specials in the GF file for use in making proofsheets via, e.g., the utility program GFtoDVI (page 323–4).

`tracingtitles`: if nonzero, strings that appear as METAFONT statements are typed on the terminal (page 187).

Neenie Billawala's article in the April 1987 issue of *TUGboat* describes how to test your printer for the best set of values for the magic numbers above. Here are some brief comments on the subject, courtesy of Rocky Bernstein, again edited by Karl:

For medium-to-low resolution devices, you can first set the `blacker` and `o_correction` to 0 and decide on a `fillin` value by looking at the diagonal of a lowercase 'z' in `cmtt10`, or various lines in LaTeX's line10 font. The diagonal should be the same thickness as the horizontal bars of the 'z'. Then I decide on the `blacker` value, generally by checking the smaller fonts for too much filling in. Finally, you can set the `o_correction` using the guidelines suggested above.

End quote. From the Docutech values we can conclude that it tends to print much lighter, it doesn't fill in diagonals as much as a Laserjet, and the overshoot is nearly ideal.

3 The proof of the pudding

To determine if the variables are set correctly you need printed copy. Unfortunately that is not possible in an article such as this. However, to proof that the different values do effect METAFONT's output we can use the MS-DOS program PK2BM[2] to convert a character from a PK font file to a bitmap. Output from BM2PK typically looks like this:

```
character : 64 (@)
   height : 29
    width : 34
     xoff : -4
     yoff : 28

  ............********............
  .........***........***.........
  ........**..............**.......
  ......**..................**.....
  .....**....................**....
  ....**......................**...
  ...**........******..........**..
  ..**.......****.....**.........**.
  ..**......***........**.......**..
  .**......***..........*****....**.
  .**.....****..........****.....**.
  **......***...........****.....**
  **.....****...........****.....**
  **.....****...........****.....**
  **.....****...........****.....**
  **.....****...........****.....**
  **.....****...........****.....**
  **......***...........****.....**
  .**.....****..........****.....*.
  .**......***..........*****....**.
  ..**......***........******....*..
  ..**.......****.....**..***...**..
  ...**.........******.....*****....
  ....**.............................
  .....**...........................
  ......**.........................
  ........**.....................****
  ..........***............******...
  ............************.........
```

It can also display bitmaps in hexadecimal format:

```
character : 64 (@)
   height : 29
    width : 34
     xoff : -4
     yoff : 28

  0007f80000
  0038070000
  00c000c000
  0300003000
  0600001800
  0c00000c00
  1803f00600
  301e0c0300
```

[2] Available at CTAN: `fonts/utilities/ps2pk/ps2pk14x/msdos/emx/pk2bm.exe` or `fonts/utilities/ps2pk/ps2pk14x/msdos/djgpp/pk2bm.exe`.

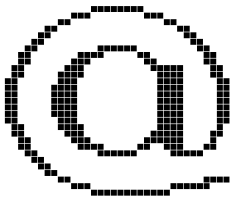
```
  3038060300
  607003e180
  60f001e180
  c0e001e0c0
  c1e001e0c0
  c1e001e0c0
  c1e001e0c0
  c1e001e0c0
  c1e001e0c0
  c0e001e0c0
  60f001e080
  607003e180
  303807e100
  301e0ce300
  1803f07c00
  0c00000000
  0600000000
  0300000000
  00c00003c0
  0038007e00
  0007ff8000
```

Even C-code can be generated:

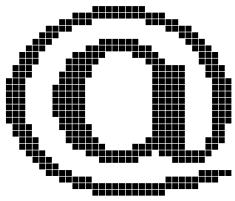
```
#define cmr5_@_width 34
#define cmr5_@_height 29
#define cmr5_@_xoff 0
#define cmr5_@_yoff 0
static char cmr5_@_bits[] = {
   0x00,  0xe0,  0x1f,  0x00,  0x00,
   0x00,  0x1c,  0xe0,  0x00,  0x00,
   0x00,  0x03,  0x00,  0x03,  0x00,
   0xc0,  0x00,  0x00,  0x0c,  0x00,
   0x60,  0x00,  0x00,  0x18,  0x00,
   0x30,  0x00,  0x00,  0x30,  0x00,
   0x18,  0xc0,  0x0f,  0x60,  0x00,
   0x0c,  0x78,  0x30,  0xc0,  0x00,
   0x0c,  0x1c,  0x60,  0xc0,  0x00,
   0x06,  0x0e,  0xc0,  0x87,  0x01,
   0x06,  0x0f,  0x80,  0x87,  0x01,
   0x03,  0x07,  0x80,  0x07,  0x03,
   0x83,  0x07,  0x80,  0x07,  0x03,
   0x83,  0x07,  0x80,  0x07,  0x03,
   0x83,  0x07,  0x80,  0x07,  0x03,
   0x83,  0x07,  0x80,  0x07,  0x03,
   0x83,  0x07,  0x80,  0x07,  0x03,
   0x03,  0x07,  0x80,  0x07,  0x03,
   0x06,  0x0f,  0x80,  0x07,  0x01,
   0x06,  0x0e,  0xc0,  0x87,  0x01,
   0x0c,  0x1c,  0xe0,  0x87,  0x00,
   0x0c,  0x78,  0x30,  0xc7,  0x00,
   0x18,  0xc0,  0x0f,  0x3e,  0x00,
   0x30,  0x00,  0x00,  0x00,  0x00,
   0x60,  0x00,  0x00,  0x00,  0x00,
   0xc0,  0x00,  0x00,  0x00,  0x00,
   0x00,  0x03,  0x00,  0xc0,  0x03,
   0x00,  0x1c,  0x00,  0x7e,  0x00,
   0x00,  0xe0,  0xff,  0x01,  0x00, };
```

Another nice program to examine or even hand-edit PK font files is the MS-DOS (or OS/2) program (PKEDITPM).[3]

A good candidate for displaying the difference between a font generated for Laserjet or Docutech is cmr5. The ampersand character will be shown because of its delicate thin lining.

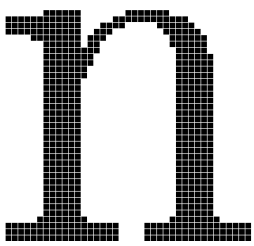


Laserjet cmr5

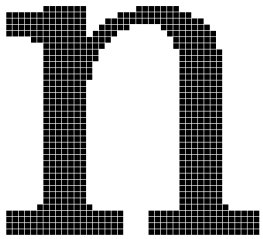


Docutech cmr5

The differences are obvious: The Laserjet font is much thinner. This was an extreme case where lines were as thin as one pixel. Naturally the differences become less and less visible with higher font sizes. At 10 pt there is still a noticeable difference as shown in the letter n:



Laserjet cmr10



Docutech cmr10

The stems of the Laserjet font are one pixel thinner, just like the serifs. At size higher than 15 pt the differences become insignificant.

4 Conclusion

It is obvious that `mode_defs` are essential for high quality output using fonts written in Metafont. If you have searched Karl Berry's `mode_def` collection and found that your device is not listed, the best thing you can do is try to find a device with similar resolution, and see if that suits. Otherwise you will have to fiddle yourself the variables mentioned. Ideally you would try normal, bold an italic versions, at sizes 5 pt, 10 pt and 15 pt.

If you make a new `font_def`, please send it to Karl Berry.[4] Please mention what fonts at what sizes you tested it on. This will help other people wondering where particular values came from.

[3] Available at CTAN: `systems/msdos/emtex/disk5/pkedit.zip` or `systems/msdos/emtex/betatest/pkeditpm.zip`.

[4] E-mail address: `karl@cs.umb.edu`.

Combining TEX and PostScript*

Vladimir Batagelj

University of Ljubljana, Department of Mathematics
Jadranska 19, 61 111 Ljubljana
Slovenia
vladimir.batagelj@uni-lj.si/vlado.html
http://www.uni-lj.si/vlado/vlado.html

March 1995

Abstract

PostScript is becoming a de facto standard as a device independent page description language. By embedding PostScript elements in TEX we can extend the use of TEX to new areas of application.
In the first part of the paper we give some general information about PostScript and its features. In the rest of the paper we present some of our own experiences and solutions in combining TEX and PostScript:

- dictionaries, prolog files and how to save a lot of space with PostScript figures produced in *CorelDRAW*, *Mathematica*, ...;
- writing TEX-PostScript macros, case: drawing graphs (combinatorics) in TEX; PostScript error handling mechanism, an application in function graph drawing macro.

Keywords: PostScript, TEX, inclusion of graphics, dictionaries, macros, error handling.

1 Introduction

Pictures, figures and color are often important elements of a document. They are foreign concepts to TEX which is essentially based on arranging and glueing of boxes.

In his *A Survey of TEX and Graphics* [6, p. 275-276] S. Rahtz discusses six approaches for producing graphics in TEX. The first five are based on the TEX system and therefore preserve device independence, but they are inflexible in those cases where a picture has to be transformed (scaled, rotated).

The sixth approach is based on the use of the TEX command `\special` with which we can include in the DVI file commands for a selected output device driver. By doing this we lose device independence; but, in the case of PostScript, and considering its graphical power and the availability of printers and previewers, this little adultery seems worthwhile. In this paper we shall take a closer look at this approach.

In the first part of the paper we give a short introduction to basic ideas and capabilities of PostScript, thus making the paper self-contained. In the rest of the paper we present some of our own experiences and solutions on PCs in combining TEX and PostScript.

2 PostScript

2.1 What is PostScript?

PostScript is a graphics programming language for describing, in a device-independent manner, text and other graphical objects and how they are placed on the page or screen.

It was developed in 1985 by Adobe Systems in a joint project with Apple Computer on the development of the Apple LaserWriter. This version is known as PostScript Level 1.

Although PostScript was initially designed as an interface between picture production and text formatting programs on one side, and printers on the other side, it evolved into a general interface language between (application) programs and display devices. Its main extensions, by different users, were:

- introduction of colors, improvements of pattern filling and halftones;
- support for composite fonts (Japanese and other Eastern alphabets);
- representation and communication of information in some computer systems — Display PostScript (NeXT, Silicon Graphics).

At the first PostScript Conference in 1990, PostScript Level 2 was announced which integrated these features into a new version of the PostScript language.

*Version: March 1995 — updated version of the paper presented at EuroTEX'94, Sobieszewo, Poland, 26-30. Sep 1994.
Math. Subj. Class. (1991): 68 U 15, 68 - 01, 68 N 15

2.2 PostScript programs and their execution

A PostScript program is a text (ASCII) file. Usually it is produced by some other graphics or text formatting program (Word, Word Perfect, *CorelDRAW*, *Mathematica*, ...), but it can be also prepared and maintained by a user and any text editor.

To obtain from a document described in TEX on *file*`.tex` its PostScript description on *file*`.ps`, we first produce, as usual, the corresponding DVI file *file*`.dvi` and translate it using some DVI-to-PS program (DVIPS, DVI2PS, DVITOPS, ...) into PostScript.

The simplest way to display the results of a PostScript program on *file*`.ps` is to send it to a PostScript printer (`copy` *file*`.ps lpt:` or `print` *file*`.ps`).

PostScript programs are either interpreted by an interpreter built into a display device (i.e. laser printer) or by a software interpreter in the user's computer. The most widespread software PostScript interpreter is Ghostscript (Aladdin Enterprises and Free Software Foundation). Ghostscript 3.12 (September 1994) implements PostScript Level 2. Ghostscript enables us to preview PostScript documents on the screen and to print them on several nonPostScript printers.

3 Basic PostScript programming

3.1 Syntax

PostScript program starts with

```
%!PS
```

followed by the description of page(s). PostScript recognizes, besides a printable subset of the ASCII character set, also characters *space*, *tab* and *newline* (CR or LF or CR LF).

Some PostScript printers use CTRL-D as an indicator of end-of-job. For this reason some application programs insert CTRL-D at the beginning of PostScript files, which is often a source of problems when we are trying to include such files in our documents.

The content of the line from `%` till the end of line is a comment.

PostScript is a stack-based language and uses a postfix (reverse Polish) notation for commands

$p_1\ p_2\ \ldots p_n\ cmd$

The interpreter puts the arguments p_1, p_2, ..., p_n on the stack and leaves the results of command cmd on it.

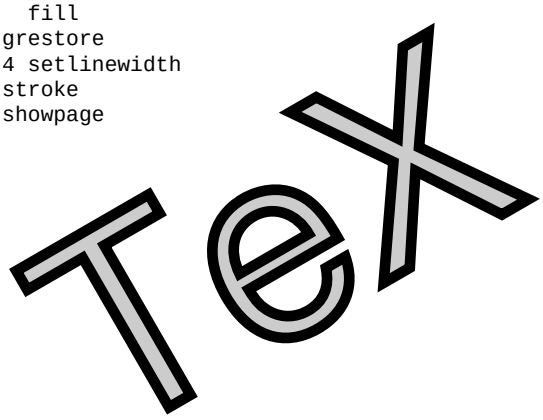
PostScript [1, 12, 2, 3, 13] is a powerful programming language which besides general programming elements: data types (integer, real, boolean, string, array, dictionary, file), control statements (`if`, `ifelse`, `loop`, `for`, `exit`, `exec`), arithmetic operations and functions (`add`, `sub`, `mul`, `div`, `idiv`, `mod`, `abs`, `neg`, `ceiling`, `floor`, `round`, `truncate`, `sqrt`, `exp`, `ln`, `log`, `sin`, `cos`, `atan`, `rand`, `srand`, `rrand`), operations and functions on other data types, conversion operators, stack commands (`dup`, `exch`, `pop`, `copy`, `roll`), environment commands(`save`, `restore`, `gsave`, `grestore`); contains also many specific graphics commands: coordinate system changing commands (`rotate`, `scale`, `translate`, `transform`), path drawing commands (`moveto`, `rmoveto`, `lineto`, `rlineto`, `curveto`, `arc`, `charpath`, `newpath`, `closepath`), attribute setting commands (`setgray`, `setcmykcolor`, `setrgbcolor`, `setlinewidth`), font commands (`findfont`, `scalefont`, `setfont`), displaying commands (`clip`, `stroke`, `fill`, `show`, `showpage`).

3.2 PostScript's coordinate system.

PostScript's own coordinate system is based on units called points (72 pt = 1 inch). It has the origin (0,0) in the lower left corner (letter = 8.5 $\times$ 11 inch = 612 $\times$ 792 pt; A4 = 21 $\times$ 29.7 cm = 595 $\times$ 842 pt). The content of the page is composed of page elements — parts of pictures or text. Each page element is determined by a set of paths (lines, arcs, curves) and their properties which are realized after the application of some displaying command. Characters are also treated as pictures, but supported by a special set of very efficient commands.

3.3 Example: Simple program

```
%!PS
/Helvetica findfont 100 scalefont setfont
40 0 moveto
30 rotate
(TeX) false charpath
gsave
  0.8 setgray
  fill
grestore
4 setlinewidth
stroke
showpage
```



The first line of the program declares that this is a PostScript program. In the second line we set the Helvetica font at size 100pt as the current font. Then we move to the point (40,0) and rotate the coordinate system through 30 degrees. In the next line we transform the text `TeX` into its outline. The command `gsave` saves the current graphic environment. We fill the interior of the outline with 0.8 gray (1 is white, 0 is black) and restore the graphical environment. Now we set the line width to 4pt and draw the outline. It has to be emphasized that path drawing and attribute setting commands create only descriptions of paths which are not realized on the page until some displaying command is issued. The command `showpage` at the end of the page requires that the interpreter display the page.

3.4 Dictionaries.

An important concept in PostScript is the notion of a dictionary. It consists of (*key, value*) pairs, which are in some sense the PostScript equivalent of the concept of a variable. The $value$ is stored under the name `/`key into the current dictionary by the command

`/`key $value$ `def`

There is a stack of active dictionaries which determine the current context. There are always two permanent dictionaries `systemdict` and `userdict` (and `globaldict`), but the user can introduce his own dictionaries.

A new dictionary of size n (number of entries) is created by the command

n `dict`

and saved in the current dictionary under the name `/`D by the command

`/`D n `dict def`

It is opened for use by the command

D `begin`

and closed by the matching command

`end`

Although dictionaries allow us to use variables in a way similar to normal programming languages, this is not in the 'spirit' of PostScript — try to do the job on the stack.

Besides data, we can store in a dictionary also procedures. Dictionaries are usually used to prepare libraries for special tasks.

3.5 User defined commands.

User defined commands (procedures) are, in PostScript, a special kind of array enclosed in braces `{ }` — executable arrays. Usually we define a procedure $proc$ by storing its body `{` $cmds$ `}` into a current dictionary

`/`$proc$ `{` $cmds$ `} def`

The following two commands define the usual units

```
/inch { 72 mul } def
/mm { 2.835 mul } def
```

The command `11 mm` puts on the stack values 11 and 2.835, `mul`tiply them and returns their product (11 mm expressed in pts) on the stack.

3.6 Example: Drawing graphs.

This example demonstrates the use of a dictionary for the simple task of drawing (combinatorial) graphs. The dictionary `Graph` contains two quantities:

`pr` – radius of a point;
`pc` – color of the interior of a point;

and four commands

r `radius` – defines/changes `pr`;
c `pointcolor` – defines/changes `pc`;
x y `p` – draws a point at (x, y);
x_1 y_1 x_2 y_2 `l` – draws a line connecting (x_1, y_1) and (x_2, y_2).

The `p` and `l` commands in the description of the graph were obtained by the *Mathematica* based system Vega [14]. The resulting graph is presented in Figure 1. Note that all lines are drawn before points.

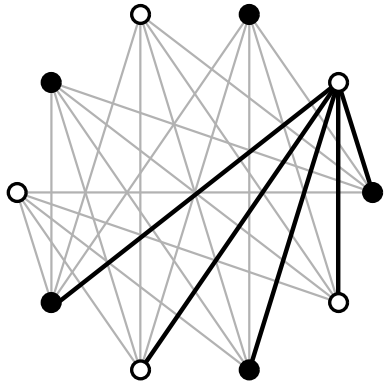


Figure 1: *Graph*

```
%!PS
%%BoundingBox: 30 30 370 370
/Graph 6 dict def
Graph begin
 /radius {/pr exch def} def
 /pointcolor {/pc exch def} def
 /p { pr 0 360 arc
  gsave pc setgray fill grestore
  stroke } def
 /l { moveto lineto stroke } def
end
Graph begin
 0.7 setgray 2 setlinewidth
 249 360  71 101 l  249 360 151  40 l
 249 360 249  40 l  249 360 329 101 l
 249 360 360 200 l
 151 360  71 101 l  151 360 151  40 l
 151 360 249  40 l  151 360 329 101 l
 151 360 360 200 l
  71 299  71 101 l   71 299 151  40 l
  71 299 249  40 l   71 299 329 101 l
  71 299 360 200 l
  40 200  71 101 l   40 200 151  40 l
  40 200 249  40 l   40 200 329 101 l
  40 200 360 200 l
 0 setgray 4 setlinewidth
 329 299  77 101 l  329 299 151  40 l
 329 299 249  40 l  329 299 329 101 l
 329 299 360 200 l
 8 radius 1 pointcolor 3 setlinewidth
  40 200 p  151  40 p  329 101 p
 329 299 p  151 360 p
 0 pointcolor
 360 200 p   71 299 p   71 101 p
 249  40 p  249 360 p
showpage
end
```

3.7 DSC — Document Structuring Conventions

PostScript program lines beginning with `%%` are special comments intended for programs (previewers, utilities) which process PostScript programs. The rules determining the structure and meaning of these comments are known as DSC — Document Structuring Conventions. A typical PostScript program structure is

```
%!PS-Adobe-3.0
%%Pages: 27
```
. . . DSC comments
```
%%EndComments
%%BeginProlog
```
. . . definitions of commands . . .
```
%%EndProlog
%%BeginSetup
...
%%EndSetup
%%Page: 1 1
%%BeginPageSetup
...
%%EndPageSetup
...
%%Trailer
%%EOF
```

The first line `%!PS-Adobe-3.0` tells us that the program conforms to DSC – version 3.0. We can omit unused DSC comments.

An example of a previewer using DSC comments is GsView (by Russell Lang) [10] which by using `%%Page:` comments allows us to see or print the selected page(s).

3.8 EPS – Encapsulated PostScript

Encapsulated PostScript format is intended to allow one to import already prepared parts of a picture into a document. The EPS file should contain only one page and shouldn't use operators that would perturb the graphics environment of the surrounding PostScript. It usually starts with a line

`%!PS-Adobe-3.0 EPSF- 3.0`

and one of the following lines should be

`%%BoundingBox:` $ll_x\ ll_y\ ur_x\ ur_y$

which defines the bounding box of the picture in the file.

4 PostScript and TEX

4.1 DVIPS and EPSF

The most popular DVI-to-PS program on PCs is DVIPS [15]. It was written by Tomas Rokicki of Radical Eye Software. DVIPS comes with two files `epsf.tex` and `epsf.sty` which contain TEX macros to include an Encapsulated PostScript graphic. It works by finding the bounding box comment, calculating the correct scale values, and inserting a `vbox` of the appropriate size at the current position in the TEX document.

Program DVIPS recognizes several forms of `\special` command. For example:

`\special{"` $cmds$`}` — includes PostScript commands in place. The user is responsible for providing space for such literal graphics. The $cmds$ are enclosed in a PostScript `save`/`restore` pair.
`\special{ps:` $cmds$`}` — inline PostScript commands — not enclosed in a `save`/`restore` pair.
`\special{header=`$file$`.pro}` — includes the contents of a prolog file $file$`.pro` in `userdict`.

4.2 PostScript-TEX packages

By the end of eighties the first attempts to combine TEX and PostScript were being made (PSLATEX(L.A. Carr), colors (F. King), pspic (K.K. Thorup), psfig (T. Darell [5]), . . .) and recently some excellent packages have been produced (changebar (J. Braams), rotating (S. Rahtz, L. Barroca), psboxit (J. Maillot, T. Sheffler), pspicture (D. Carlisle), epsfig (S. Rahtz), geom (S. Levy [11]), foiltex (J.L. Hafner [8]), seminar, PsTricks (T. Van Zandt [18, 16, 17]), PSNFSS (S. Rahtz)). In the new LATEX there are some PostScript based packages: pict2e, graphics and color [9].

4.3 Epsfig

The most popular package for inclusion of EPS figures in TEX is epsfig (S. Rahtz, based on epsf and on T. Darell's psfig). It extracts the bounding box information from the file and positions the figure according to the user's wishes on the page. Its main macro has the following parameters:

`\epsfig{file=`*file*`,height=`h`,width=`w`,%`
`  clip=,rotate=`a`,silent=,%`
`  bbllx=`lx`,bblly=`ly`,bburx=`ux`,bbury=`uy`}`

5 PostScript, TEX and other programs

5.1 CorelDraw 3

After we enter *CorelDRAW* we can first determine the dimensions of the picture by selecting in menu `File/Page Setup...` the option `Custom`. This enables fields for user's settings. First, if necessary, we change the units of measurement — for example to `millimeters`. Afterwards we enter the selected width and height of the picture and confirm our decision with a click on the `OK` button. Then we draw a picture. For possible future changes we save it in *CorelDRAW* format (options `File/Save As...` and `File/Save`) on *file*`.cdr`.

To get the picture in EPS format we enter the menu `File/Export`. In the pop-up submenu `List Files of Type:` we select the option `Encapsulated PostScript, *.EPS`. Then we move in the directory submenu to the directory where we keep the pictures, enter the name of the file, and confirm our selections with `OK`. A new dialog box `Export EPS` appears in which we select the option `Image Header/None` and confirm it with `OK`. The picture is saved in EPS format in the file *file*`.eps`. The bounding box of the picture is determined by *CorelDRAW* from the picture. Such pictures can be easily included into our TEX document:

```
\documentstyle[11pt,epsfig]{article}
\begin{document}
...
\begin{figure} \begin{center}
\centerline{\epsfig{figure=encormix.eps,width=70mm}}}
 \caption{Clip-art from \CD \label{CD}}
\end{center} \end{figure}
...
\end{document}
```

or in LATEX2e [9]

```
\documentclass[11pt]{article}
\usepackage[dvips]{graphics}
\begin{document}

...

\begin{figure} \begin{center}
 \resizebox{70mm}{!}{\includegraphics{graph.eps}}
 \caption{Clip-art from \CD \label{CD}}
\end{center} \end{figure}

...

\end{document}
```

In Figure 2 a clip-art composition, made in *CorelDRAW* in few minutes, is presented.

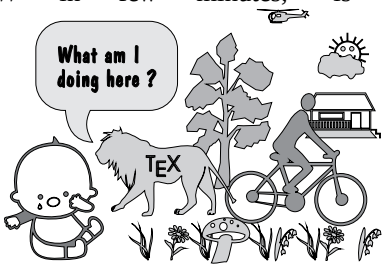


Figure 2: *Clip-art from CorelDRAW*

When we have several *CorelDRAW* figures to be included in our text we can achieve substantial savings both in disk space (15K per file) and processing time if we notice that all *CorelDRAW* EPS files have large parts that are identical. The structure of an EPS file produced by *CorelDRAW* 3 is as follows

```
%!PS-Adobe-2.0 EPSF-2.0
%%BoundingBox: 7 6 191 139
%%Creator: CorelDRAW!
%%Title: TESTA.EPS
%%CreationDate: Sat May 14 21:20:40 1994
%%DocumentFonts: AvantGarde-Book
%%DocumentProcessColors: Cyan Magenta Yel...
%%EndComments
%%BeginProlog
/AutoFlatness false def
% -------------- POSTSCRIPT PROLOG FOR CO...
% Copyright 1992 Corel Corporation.  All ...
/wCorelDict 300 dict def wCorelDict begin...

...

%%EndProlog
%%BeginSetup

...

%%Trailer
 end
```

The contents of the file after the `%%BeginProlog` until the `%%EndProlog` is the same for all *CorelDRAW* EPS files. It starts with a dictionary

```
/wCorelDict 300 dict def wCorelDict begin
```

which is ended by the `end` in the trailer at the end of the file.

We can save the constant contents between the `%%BeginProlog` and the `%%EndProlog` as a header file `corel3.pro` (15K), which is read only once by a style file `corel3.sty`

```
\long\def\ifundefined#1#2#3{\expandafter
 \ifx\csname#1\endcsname\relax#2\else#3\fi}
\ifundefined{Corel3DrawSTY}
 {\def\Corel3DrawSTY{}}{\endinput}
\immediate\write16{Document Style Option %
 'CorelDraw 3' ver 1.0 / 14-May-94 / VB}
\special{header=corel3.pro}
```

We have to insert in `corel3.pro` an `end` a line before the `%%EndProlog`.

Then we can in each *CorelDRAW* EPS file delete the contents between the `%%BeginProlog` and the `%%EndProlog` and replace it with a command `wCorelDict begin` thus obtaining a shortened file *file*`.cps`. This can be done manually using an editor or by a simple program.

We can still view the shortened files by Ghostscript requesting

`gs corel3.pro` *file*`.cps showpage.ps`

On CTANs we can get also a PostScript version of standard TEX fonts. If we register (some of) them with ATM (Adobe Type Manager) they become available to *CorelDRAW* and we can use them for labels in our pictures (see Figure 3).

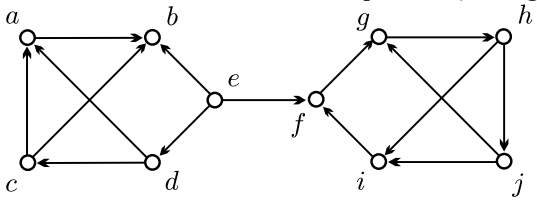


Figure 3: *Picture from CorelDRAW with* `cmmi10` *labels*

5.2 CorelDraw 5

Essentially the same applies as for *CorelDRAW* 3. There are some differences in the preparation of CPS files. We extract (once) from a *CorelDRAW* 5 EPS file the segment from `%%BeginProlog` till `%%EndProlog` (including) to the file `corel5.pro` (21.4K). In this file we insert after `wCorel5dict begin` the command

```
/AutoFlatness true def
```

We produce a CPS file by deleting from the corresponding EPS file the segment from the first `%%BeginSetup` till `%%EndProlog` (including).

For unknown reasons in *CorelDRAW* 5 the origin of coordinate system is placed at position around $(2200, 2200)$. To preview the *CorelDRAW* 5 CPS files with Ghostscript we write on the file `cd5.ps` the lines

```
%!PS
-2200 -2200 translate
```

Now we can require

`gs cd5.ps corel5.pro` *file*`.cps showpage.ps`

5.3 Mathematica

Using *Mathematica* we draw a picture and save it with command `Display["`*file*`",`*picture*`]` to the file *file*. For example:

```
d:\>math
In[1] := <<Graphics‘Polyhedra‘
In[2] := Show[Graphics3D[Dodecahedron[],
 ColorOutput->GrayLevel,Boxed->False]]
In[3] := Display["dodec.ps",%]
In[4] := Exit
```

The saved *file* contains a PostScript description of a picture. After we exit *Mathematica*, we have still to transform the saved picture into EPS format. This can be done with the DOS command `eps` *file EPSfile*. In our example

```
d:\>eps dodec.ps dodec
```

The command `eps` is determined by a batch file `eps.bat` which contains a call to *Mathematica* program `rasterps`

```
rasterps -format eps -file %2.eps %1 %3 %4 %5
```

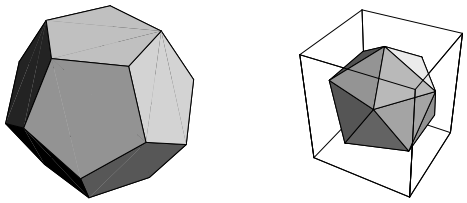


Figure 4: *Pictures from Mathematica*

This program inserts at the beginning of the *file* some DSC comments and the *Mathematica* prolog.

```
%!PS-Adobe-2.0 EPSF-2.0
%%BoundingBox: 0 0 138 138
%%Creator: Mathematica
%%CreationDate: Thu Aug  4 03:01:20 1994
%%EndComments
%%BeginPreview: 200 200 1 200

...

} bind def
%!
%%Creator: Mathematica
%%AspectRatio: 1.00154
MathPictureStart
%% Graphics3D
/Courier findfont 10  scalefont  setfont
```

...picture

```
% End of Graphics
MathPictureEnd
end
showpage
```

thus producing an EPS file.

Again we can input *Mathematica* prolog (24K) only once and insert at the begining of the *file* a short header

```
%!PS-Adobe-2.0 EPSF-2.0
%%BoundingBox: 0 0 276 276
%%Creator: Mathematica
% *** Partial EPS form produced by PS4TeX
%%CreationDate: ...
%%EndComments
Mathdict begin
```

and add as a last line `end showpage`. For unknown reasons the bounding box provided by `rasterps` is only one half of the correct one.

For preparing `cps` and `mps` files, a short program `PS4TEX` was written. Its latest version for the PC is available in self-extracting format by anonymous FTP from

```
uek.uni-lj.si:pub/vlado/tex/ps4tex*.exe
```

6 TEX-PostScript macros

As already said, in principle PostScript programs are generated by other programs which are also responsible for their correctness. Different approaches to this problem were used in the existing TEX-PostScript packages (see, for example, Van Zandt's PSTricks and Carlisle's pspicture).

Although PostScript is a complete programming language, the generating program should perform all computations and other processing that it can. Nevertheless, since TEX lacks (trigonometric and other) functions, we sometimes leave PostScript to do the job.

6.1 Error mechanism in PostScript

The dictionary `systemdict` contains as its entries two dictionaries related to error handling. The `errordict` contains built-in procedures for all possible error types and a procedure `handleerror`. On an error the PostScript interpreter executes the corresponding error procedure which saves the information about the error in the second dictionary `$error` and executes the command `stop`.

We can catch a `stop` inside the commands context $cmds$ by the command

{ $cmds$ } `stopped`

which, when an error occurs, pops the corresponding part of execution stack and returns a boolean value `true`.

The command `stop` transfers control to the innermost `stopped` context. The main interpreter loop is always such a context and invokes the standard `handleerror` procedure as a part of error recovery. The user can replace this procedure by his or her own procedure.

6.2 Example: Functions

The following simple PostScript program draws (see Figure 5) a function $f(x) = 200 \sin 2x$.

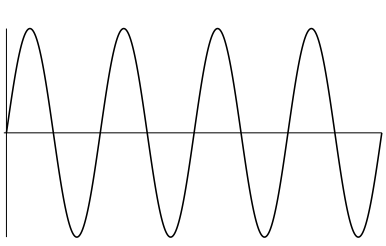


Figure 5: *Function*

```
%!PS
%%BoundingBox: -5 90 720 510
0 300 moveto
0 1 720 { %     from step to
  dup     %     x x
  2 mul   %     2*x
  sin     %     sin(2x)
  200 mul %     200 sin(2x)
  300 add %     300 +
  lineto  %
} for
3 setlinewidth stroke
-5 300 moveto 720 300 lineto
0 100 moveto 0 500 lineto
1 setlinewidth stroke
showpage
```

But, if we try to draw, by inserting a `sqrt` after the `sin`, a function $f(x) = 200\sqrt{\sin 2x}$ the error `rangecheck` occurs. We can overcome the problem (see Figure 6) by intercepting the error by the command `stopped` as shown in the improved version of the program

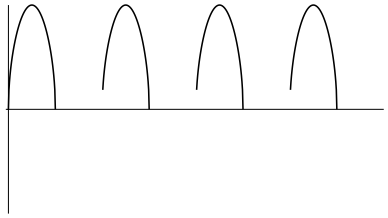


Figure 6: *Function*

```
%!PS
%%BoundingBox: -5 90 720 510
/up true def
0 300 moveto
0 1 720 {
  { dup 2 mul sin sqrt 200 mul
    300 add } stopped
  { pop /up true def }
  { up
    { moveto /up false def }
    { lineto }
    ifelse }
  ifelse }
for
3 setlinewidth stroke
-5 300 moveto 720 300 lineto
0 100 moveto 0 500 lineto
1 setlinewidth stroke
showpage
```

There is still a lot of work to convert this solution to a macro for drawing functions, but the details go beyond the scope of this paper.

7 Conclusion

Combining TEX and PostScript returns TEX into the competition with other typesetting systems. It will be a big challenge for the TEX community in the following years to provide the support for this approach.

References

[1] Adobe Systems Inc.: PostScript Language, Reference Manual. Second Edition. Addison-Wesley, Reading, MA, 1990.

[2] Adobe Systems Inc.: PostScript Language, Tutorial and Cookbook. Addison-Wesley, Reading, MA, 1985.

[3] Adobe Systems Inc., Reid G.C.: PostScript Language, Program Design. Addison-Wesley, Reading, MA, 1988.

[4] Aladdin Enterprises: Ghostscript, `use.doc`. 30 Sep 1994. `ftp.cs.wisc.edu:pub/ghost`, 1994.

[5] Darrell T.: Psfig/TEX 1.10 User's Guide. `whitechapel.mit.media.edu:pub`, 1994.

[6] Goossens M., Mittelbach F., Samarin A.: The LATEX Companion. Addison-Wesley, Reading, MA, 1994.

[7] Goossens M., Van Herwijnen E.: Scientific Text Processing. International Journal of Modern Physics C **3**(1992)3, 479-546.

[8] Hafner J.L.: Making Foils Using FoilTEX. 21 aug 1992. IBM Almaden Research Center, San Jose, CA.

[9] Lamport L.: A Document Preparation System LATEX. User's Guide and Reference Manual. Second Edition. Addison-Wesley, Reading, MA, 1994.

[10] Lang R.: Gsview, `readme.doc`. 9 Dec 1994. `ftp.cs.wisc.edu:pub/ghost/rjl`, 1994.

[11] Levy S.: The `geom` style for LATEX. Geometry center, University of Minnesota, `geom.umn.edu`, July 1992.

[12] McGilton H., Campione M.: PostScript by Example. Addison-Wesley, Reading, MA, 1992.

[13] Monsarrat J.: — PostScript — Answers to Questions. The `comp.lang.postscript` FAQ v2.2 (12-26-1993). `wilma.cs.brown.edu:pub`, 1993.

[14] Pisanski T.: Vega 0.3 alpha release, 1994. `http://vegaj.mat.uni-lj.si/vega03/`.

[15] Rokicki T.: DVIPS: A TEX Driver. Manual, v5.521. `labrea.stanford.edu:pub`, 1994.

[16] Van Zandt T.: PSTricks, PostScript macros for Generic TEX. User's Guide. Version 0.93a, `princeton.edu:pub/tvz`, 12 Mar 1993.

[17] PSTricks et Seminar (D. Girou, ed.). Cahiers GUTenberg 16, 1994. (in French)

[18] Van Zandt T.: seminar.sty, A LATEX style for slides and notes. User's Guide. Version 0.93, `princeton.edu:pub/tvz`, 16 Feb 1993.

[19] Walsh N.: Making TEX Work. O'Reilly, Sebastopol, CA, 1994.

See also:

```
http://www.cs.wisc.edu/~ghost/index.html
http://www.cs.indiana.edu/docproject/
     programming/postscript/postscript.html
http://www.adobe.com/
http://www.ucc.ie/info/TeX/tug/
http://www.tex.ac.uk/UKTUG/home.html
http://www.stat.wisc.edu/latex.html
http://info.desy.de/UCO/latex2e.html
```

8 Acknowledgements

Supported in part by the Ministry of Science and Technology of Slovenia.

Portable Documents: Why Use SGML?*

David Barron

Department of Electronics and Computer Science,
University of Southampton,
S09 5NH Southampton, England
dwb@ecs.soton.ac.uk

1 Introduction

In this article we present a few ideas as a framework for the discussion of portable documents. We address a number of questions:

- What are portable documents?
- Who needs them, and why?
- How to produce them, now and in the future

2 Documents

Traditionally, a document was a file (or a deck of cards), and consisted solely of text. Today, documents are typically *compound*, a mixture of text and graphics (bit-map or line art) that can be rendered on paper or screen. Additionally, they may include hypertext links (in which case they can only be viewed on screen). A recent development is the ability to incorporate video and sound in a compound document, either embedded within the document or linked by a pointer: such a document is a *multimedia* document. Hypertext-style links may also be included to form a *hypermedia* document: evidently, multimedia and hypermedia documents can only be 'read' on a suitably equipped computer system.

World Wide Web (WWW) documents are a special case of compound hypermedia documents where the links are to other documents elsewhere on the Internet. They can be regarded as virtual documents, in the sense that the whole document never exists as a single identifiable object. More generally, we can define a *virtual document* as a structured collection of information from which instances of documents and other resources can be derived. Examples include:

- The Oxford English Dictionary which exists as a database from which are derived various printed editions (Shorter, Concise, Pocket etc.), as well as the CD-ROM version.
- Critical editions of a literary text, where a single source 'document' contains all the variations, and can be printed out using different variants as the base text.

3 Portability

The definition of portability that we shall use in this discussion is the ability to transmit the document digitally (over a network, or on a disk or CD-ROM) and re-create a faithful rendering of the document after transmission, if need be on a different hardware and/or software platform from that on which the document was originally created. It is important to observe that there are three different forms in which the text and graphics in a document might be re-created:

- with absolute visual fidelity,
- with approximate visual fidelity,
- retaining content only.

4 Who needs portable documents, and why?

Three different needs for portable documents can be adduced

1. Publishers need them in order to distribute electronic books and journals.
2. Communities with common interests who need to share information need them. An example is a scientific research community whose members use diverse hardware and software.
3. Librarians responsible for digital archives need portable documents, since they cannot assume that a particular hardware/software platform will exist in perpetuity.

5 Examples of successful portability

- Computer science researchers and software manufacturers distribute documents as PostScript files. This works well if the fonts employed are restricted to the basic 35, and the use of Adobe Acrobat (pdf files) increases portability when other fonts are used.
- The Physics pre-print library at Los Alamos National Laboratory is used by many physicists world-wide: over 10,000 retrievals per day are reported. The archive holds pre-prints in LaTeX and PostScript formats (figures in PostScript only). This is successful because the Physics community has for some years used TeX as its preferred means of exchanging information.

*Reprint from the Annals of the UK TeX Users Group **Baskerville**, Volume 5.2, March 1995. Published with permission of both Baskerville editor and author. Presented at the UK TeX Users Group conference 'Portable Documents: Acrobat, SGML, and TeX', on 19 January 1995, London, England.

- WWW documents are highly portable, since their rendering is (almost entirely) determined by the browser software, and the use of a common mark-up language (HTML) ensures portability.

6 Achieving portability

At first sight it appears that portability might be achieved by agreeing standards (e.g. LaTeX, PostScript, ODA, HTML). At present there is too much choice, and no obvious winner, especially in hypermedia documents. This is a sign of an immature technology. Another important fact to take into account is that it is difficult to impose standards in some environments (e.g. academia) where personal preferences lead to the equivalent of religious wars.

Particular problems in achieving portability arise from varying fonts and character codes e.g. in handling European languages. Unicode will go a long way towards solving the character codes problem.

7 Why use SGML?

SGML provides a formal and portable definition of document structure. SGML syntax can define a hierarchical structure of embedded document parts, and can associate a type with each component in the hierarchy. By associating a rendering definition with each type of component, it is possible to achieve a portable document. In particular, SGML provides a uniform archive format for a library of portable documents.

An example

Suppose it is required to maintain a library of technical documents in an environment where some authors use LaTeX, whilst others use Microsoft Word. We can define an SGML DTD for the document structure, together with LaTeX and Word styles to define the rendering. This opens up three possibilities:

1. Author in SGML and use a tool to produce a LaTeX or Word version from which the printed version can be produced
2. Author in LaTeX and use a tool to translate to SGML to produce the archive copy
3. Author in Word and use a tool to translate the RTF form to SGML to produce the archive copy.

In addition to the SGML version of the documents, the archive must contain the Word and LaTeX style files and the translation tools. Once this is done, anyone can collect a document, the required style files and tools and produce a copy of the document. This will of course only work for text documents. For any document with graphics content, and for hypermedia documents, more is required. This is possible in principle, but much remains to be done.

8 The future

A combination of SGML and OpenDoc is probably the best way forward. OpenDoc provides an architecture for portable documents: it treats a document as a container for a collection of 'parts', each of which can have other parts embedded within it. Each type of part has associated programs to edit and render it, so that documents can be re-created with varying degrees of fidelity depending on the availability of rendering software for the particular varieties of parts that it includes.

OpenDoc is a dynamic architecture, and assumes that a new type of part may occur at any time. In principle SGML can be used to describe the static structure of an OpenDoc document, providing the final link in the portability chain.

Formatting SGML Manuscripts*

Jonathan Fine

203 Coldhams Lane,
Cambridge CB1 3HY, England
j.fine@pmms.cam.ac.uk

1 Formattting SGML Manuscripts

This article is about typography, SGML, TEX, and SIMSIM, which is a new TEX macro package. Close by are copies of several of the OHP transparencies. They were typeset *directly from an SGML document instance* using SIMSIM.

First some words about the title slide. Documents can be formatted for several purposes. They may be typeset for printing, or for conversion to Adobe PDF format. They might be formatted for viewing on a computer monitor, as is done by the WEB browsers for HTML. They might be formatted for display and alteration by a visual or WYSYWIG editor. Formatting is the process of supplying fonts, dimensions, line and page breaking rules and so forth, so as to produce a representation of the document that is (we hope) well adapted to the display medium and the needs of the user. Rendering will convert this formatted document into bitmaps or whatever that can be displayed or printed.

In my opinion SGML is as important for structured documents as ASCII is for character sets (and SQL is for databases). It is the standard that will allow different machines and different software programs to share documents. In the title I use the words 'manuscripts' to emphasise that my focus is on human communication from author to reader, and not transference of bytes from one machine to another. Human beings have special qualities, which can be reflected in the manuscripts they produce. More on this later.

Still on the title slide, the subtitle 'Much Ado about Nothing' has two meanings. The first is that in five to ten years the formatting of SGML manuscripts will be no big deal, just as today PostScript is nothing very special. The second is that success requires taking pains or 'making much ado' over the spaces. Which brings us on to the second slide.

2 Spaces Between Words

Typography is not the only art where a sound sense of space is vital. Architecture and music are others. The quotation from Schnabel expresses my view beautifully. It is one thing to get the fonts and sizes right (to play the notes on the score) and another to get the little pauses or spaces right, and also the timing of the line and page breaks. In *The TEXbook*, Knuth quote Jan Tschichold 'Every shape exists only because of the space around it. . . . Hence there is a 'right' position for every shape in every situation. If we succeed in finding that position, we have done our job.' Much of the typographic art involves getting the space right. Getting the choice of fonts right is another skill.

Even if we cannot reach the subtle virtues just expressed, we should strive to avoid gross errors. I'm sure we have all seen two words on a page with an extra space between them, as compared to their neighbors. Often this happens because the author has for some reason placed two spaces between the words (this is the sort of things that humans are good at doing) both of which have been treated as significant by the subsequent processing. TEX's default reading rules automatically solve most of these problems, but not when braces for emphasised text and the like are present.

The writing of this article (in LATEX) provided an example of this. In an earlier version I had written

```
\subsection*{ Who owns what?}
```

and the like to begin subsections. This results in an unwanted space at the start of the title. Like so:

3 Who owns what?

The making of books involves lots of co-operation, and the participants benefit when there are clear boundaries and responsibilities. For example, many authors expect their spelling and punctuation to be corrected during the publishing process, but object to their words being otherwise changed. Newspaper journalism necessarily has different rules, as does academic journal publishing. But as a general rule the author supplies the words, the formatter the spaces. Problems arise if the author has control over spacing, or fonts for that matter. During production copy-editing and other changes will be made to the author's words. If supplied as a computer file, the author can reasonably expect to be sent back another computer file just like the one that was sent in, but containing the words as actually printed. This returned file should not exercise any control over the spaces between words, for neither did the author's original file.

*Reprint from the Annals of the UK TEX Users Group **Baskerville**, Volume 5.2, March 1995. Published with permission of both Baskerville editor and author. Presented at the UK TEX Users Group conference 'Portable Documents: Acrobat, SGML, and TEX', on 19 January 1995, London, England.

Punctuation is a great problem. By and large, the author should supply the correct punctuation mark or logical structure. The formatter must choose the font and the spacing around the punctuation mark. This will depend on the rules of style required by the publisher. So at least three parties are involved. Should the design or rules or style used by the formatter be changed, so may the punctuation marks used. The more that can be programmed into the software, the less need there is for human action. There will always be exceptions. Production staff will need on occassion to impose their will on the software's production of the formatted document.

4 Manuscript Problems

These will, in an ideal world, never arise. In an ideal world others do all they can to prevent or solve your problems. And we do all we can to help others. In reality the author might be preparing the manuscript using an ordinary text editor, or a word-processor with an SGML add-on. There are likely to be stray spaces and carriage returns scattered across the file. There might even be space between the last word of a sentence and the closing period or other punctuation! Particularly if an end-tag intervenes. If not ignored, if they influence the final printed page, then a few authors will discover and use this feature. Others will be distracted from the writing of words by the need to get the spaces 'right'. But we have agreed that the spaces belong to the formatter. Thus, the three 'Hello world' messages should be formatted identically. To do otherwise is to allow the author power over spacing.

For most elements it is reasonable to assume that their boundaries do not divide words. And also that between words a space should be supplied. Thus, each line in the displayed nursery rhyme should be formatted in the same way. The formatter should ignore 'extra' spaces and supply those that are 'missing'. More subtle is this. What is the natural size of space to provide between a bold word and a word in the default (say roman) font? This is a typographic question, and so has nothing to do with which element (if any) the space character appears. Should it be a bold-sized space, a roman-sized space, the larger, some average, or some other value.

(The OHPs have been set, for simplicity, with a space between characters depending only on current font size, but not font style. Where speed is more valuable than typography, as when an author is writing the words of a manuscript, or when the display device is a computer monitor incapable of subtle expression, this is the right choice. A quality publisher might wish to specify more closely the interword spacing.)

The thrust of this slide is that the formatting process cannot assume that the input file is 'just so' and correct for the intended processing. More likely it is an electronic manuscript, with electronic analogues to the physical imperfections that paper manuscripts present. We do hope however that it can be read.

5 What is TEX the program?

TEX is the portable program *par excellence*. It also has very few bugs. It is stable across time. It has an ethos different from commercial software, which often charges maintenance for bug reports to be responded to. With TEX one is given a modest monetary reward for finding a bug.

It is worth remembering that LATEX is not only a macro package but also an input file syntax. Because TEX is programmable, no fixed input syntax is required. Given sufficiently tricky macros, the mighty lion that is TEX can be made to imitate other beasts, such as the unforgetting elephant that is SGML. SIMSIM is just such a set of macros.

(The usual TEX approach, when confronted with SGML files to typeset, is to translate into LATEX or the like before calling on TEX to do the typesetting. However, it seems to me that this approach cannot but fail to give the author control over spacing, and to mishandle manuscript problems, unless the translation process is extremely sophisticated. It will need to know about the typography intended for each element and also the character data attributes. Add to this the legendary problems LATEX has with verbatim in titles and so forth, and the limitations should become apparent. Translation to LATEX might have been the best there was available, but it is certainly not the best that is possible.)

6 What is simsim?

This brings us to the final part of the talk, which is a software announcement. The OHPs were typeset using a preliminary version of a TEX macro package SIMSIM that I have been developing for several years, and which is close to completion. The English word 'sesame' is already a registered computer software trademark, so I have chosen to use the Arabic word 'simsim'. Both are descended from an Akkadian word, current in Mesopotamia at least 4500 years ago. Simsim is one of the oldest words known to humanity. It is also the key in the classic story of Ali Babar.

There are two sides to SIMSIM. Input and output. Input is SGML and also style files. Output is pages formatted by TEX. The title slide of the talk was typeset from:

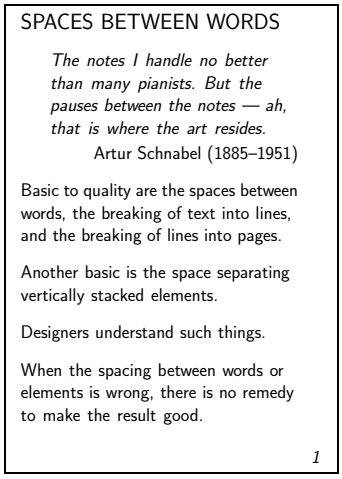
```
<title-page title =
"FORMATTING / &SGML / MANUSCRIPTS /
 - or - /
MUCH ADO / ABOUT /NOTHING"
>

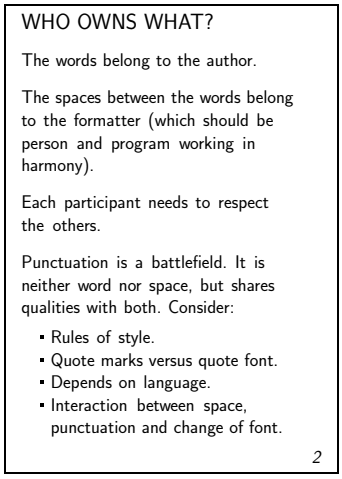
<par> UKTUG and BCS-EPSG meeting </>

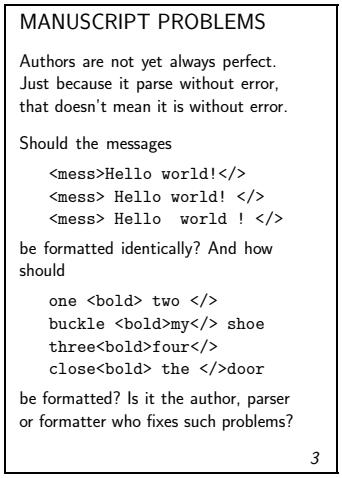
<ol>
<li> (c) Copyright 1995 </>
<li> Jonathan Fine </>
<li> 203 Coldhams Lane </>
<li> Cambridge </>
<li> CB1 3HY </>
</ol>

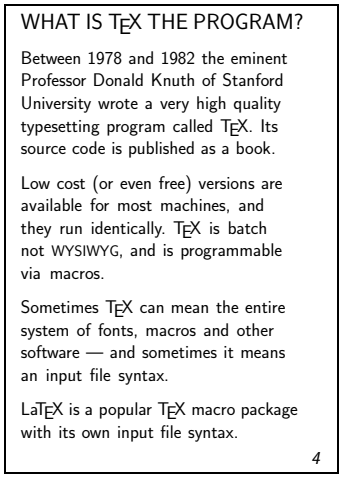
</title-page>
```

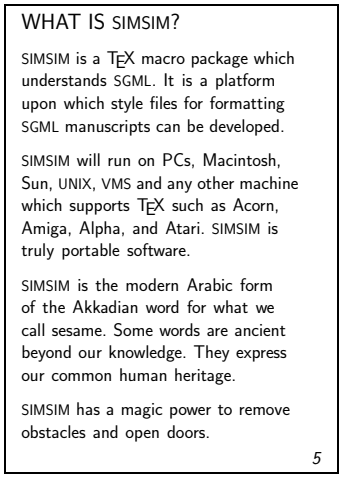
Notice that the title has been entered an attribute value, with the line breaks denoted by forward slash '/' or solidus characters. This is a notation in wide use for displaying

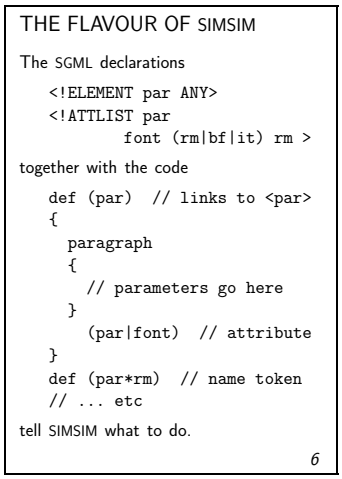


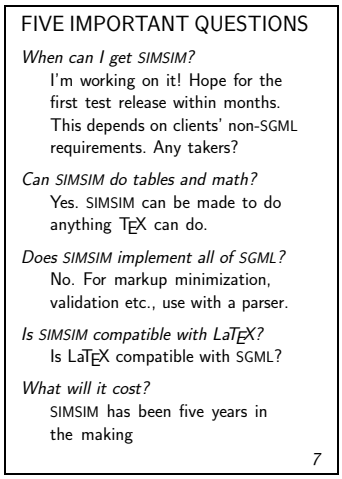












line breaks in verse quoted as flowing text within a paragraph. Suppose one were presented with the title slide and were asked to encode as an SGML element. This is the sort of thing that the Text Encoding Initiative Guidelines were developed for. One would record that it was a title page, that such and such was the title text, and so forth. It is this approach that led me to use the solidus to denote line breaks in the title text. This then is the sort of input manuscript that SIMSIM will be dealing with. Note that the formatter has not been misled by the irregular spaces in the title attribute value. The `&SGML` is an entity refence. In the title it produces itself in the current font, but elsewhere it is appearing in a smaller font. This is done using TEX's macro capabilities.

7 The Flavour of simsim

The parsing of an SGML manuscript makes the data within it available to the formatting (or whatever) application. There is even a specification (the Element Structure Information Set) of what data is available and when. Built into SIMSIM is an SGML parser. Writing a SIMSIM style file

is a matter of linking TEX actions to SGML events, such as the parsing of a start tag. The less technically minded might like to skim the following description as to how this is done.

Another part of SIMSIM is an enhanced programming environment for the writing of TEX macros and SIMSIM style files. Within a SIMSIM macro file the characters `(par)` denote a token that is called at the end of the parsing of a `<par>` start tag. It is up to the application or style file to define this token to perform the required actions.

Start tags can carry attributes. The characters

```
(title-page|title)
```

in a SIMSIM file represent a control sequence whose expansion is the text read by the parser as the value of the (character data) attribute `title` of the `title-page` tag. It is then up to the style file to typeset this data, or to write it to a file, or to otherwise dispose of it.

The other main type of attribute is the name-group. Loosely, this corresponds to the 'radio buttons' that graphical user interfaces provided. Each such attribute has a short finite list of possible values. For example, the HTML `IMG` tag has an `ALIGN` name group attribute, whose values can be `top`, `middle`, or `bottom`. Because SIMSIM incorporates an SGML parser, the style file need not worry about getting this information. Indeed, great errors are liable to occur if it attempts to do so. Rather, the parser makes this data available for the application to use.

For example, with the HTML `ALIGN` name group attribute the process goes like this. Within the SIMSIM programming environment the characters

```
(img|align)
```

represent a token whose expansion will be set by the parser to be one of

```
(img*top)
(img*middle)
(img*bottom)
```

according to the option selected by the author of the manuscript. The style file should assign appropiate values to the three tokens above, for example

```
let (img*top)    = vtop
let (img*middle) = vbox
let (img*bottom) = vcenter
```

(these are illustrative values, and are not necessarily sensible) and then

```
(img|align)
{
  //  the image goes here
  ...  ...   ...
}
```

will cause the image to be processed in accordance with the attribute value specified in the manuscript. This is all rather easier to do than to explain. Similar mechanisms are provided to link actions to `SDATA` entitities.

The observant reader may notice that I have played fast and loose with the case of tag and attribute names. For the reference concrete syntax (used by almost all SGML applications) these names are to be converted to uppercase when read. (This is controlled by a parameter in the SGML declaration.) This is in practice quite important, and so SIMSIM converts to uppercase when it parses tag and attribute names, and the same with the programming environment.

8 Five Important Questions

This slide is my attempt to anticipate the questions the audience would like to ask. (The untechnical should stop skimming.) To amplify my answers, I am looking for SGML-aware TEX users who would like to be early users of SIMSIM. Tables and math capabilities will, I hope, be developed to meet customers specific needs. I do not think it best that I try to anticipate their requirements. So much will depend on the SGML DTDs they use, or intend to use. Please contact me if you have any specific questions, and particularly if you are interested in being a test site.

At the meeting I was asked some good questions. Firstly, it is possible to have the processing attached to a tag depend on the context? The answer is yes. For example, the bulleted items on slide two are `<li>` elements, as on the title page, but within a `<bl>` rather that `<ol>` list. This is because the action attached to a tag is held as a TEX control sequence token, whose meaning can be changed just like any other control sequence. So the token represented by `(bl)` can change the meaning attached to `(li)`. (In fact this may not be the best method, there are other ways.)

Another question was how does it relate to LATEX? So far as I am concerned there is no relation with LATEX, and no means of converting documents from one form to another. Or style files for that matter. SIMSIM and LATEX both start with uninitialised TEX, but from there proceed in different directions and with different assumptions. I don't see any interaction between the SIMSIM and the LATEX worlds, and if somebody creates one, that's not my doing. A related question (motivated by legacy documents perhaps) is whether, if you have well structured TEX documents, you can get something like SGML out of it. My answer is that probably you can, but that is not the problem I set myself, and not a problem I have plans to solve.

Performance was another question. How long would it take to process a long document? This depends on the computer one has, and on the mix of text and markup in the document. Preliminary tests indicate the same order of speed as LATEX. And do I have a manual? At the moment it's not developed to such a point that I can offer manuals. But I'd like to. I want it to be a proper product. At this point it is in the process of development and I'm looking for clients who'd like to take some risk with me, or at least make some effort. I also want to supply support. Further to that, I was asked, will I be offering maintenance costs (the usual commercial practice) or rewards (Knuth's practice with TEX)? After the laughter had died down, I declined to answer the question, explaining that I did need to earn money. This was the last question.

SGML and LaTeX*

Horst Szillat

Sella-Hasse-Str. 31,
D–12687 Berlin, Germany
szillat@berlin.snafu.de

SGML — *Standard Generalized Markup Language* — is a formal language to describe structured text documents. It should be introduced here by comparison to TeX and LaTeX.

It is interesting to have a look at how Donald E. Knuth introduces TeX in the TeXbook himself. The beginning is to simply type in the text and TeX mainly does what one expects it to do. Quite a lot of more or less complex rules have been implemented to provide these results. An example of this behaviour is the *space factor code* (`\sfcode`). Using this code TeX is able to identify most of the ends of sentences. Moreover TeX is realized in a way that one can program almost all kinds of printing layouts. In this way one can program a macro which influences the layout in any place. So TeX is a layout oriented system which is able to format texts for printing and to do a bit more.

Although LaTeX simply is TeX, too, and has all these characteristics, too, it introduces a new idea of representing the text input. The basic idea is that a text is given in the form of *embedded environments*. The layout of a text portion depends on the environment it is embedded in. Moreover, the layout of whole environments may depend on which other environment they are embedded in. The user can define new environments (`\newenvironment`) which realize a user defined layout. But the main point is that the author inputs his text on a less technical but a more abstract level. This way LaTeX enforces the idea of separating the text structure from the printing layout. Changing the layout in LaTeX means to replace the existing style files, only. One could do the same in plainTeX directly, of course. One can do structured programming in assembler, too, but assembler does not enforce it.

Now one can simply say SGML is LaTeX without TeX to be written in a slightly different manner. This means SGML is a representation of the text in its hierarchical structure without any idea of a layout. If one has lost the layout there has to be an advantage on the side of the text structuring. And so it is, indeed. LaTeX's environments are called *elements* in SGML. Within a certain model one can now define which way the elements are embedded in each other and where text is to be allowed. Within that model the amount and the order of embedded elements and text is defined.

Such a definition of a text structure is called *document type definition* (*DTD*). The 'best-known' example of a SGML document type definition is HTML (*Hypertext Markup Language*) used for the World Wide Web. While processing the document an SGML-parser is able to validate the structure of the document by the given document type definition. A simple example should illustrate this:

```
<!ELEMENT section    - - (paragraph?,subsection+)>
<!ELEMENT subsection - - (paragraph,paragraph+)>
<!ELEMENT paragraph  - - (#PCDATA)>
```

These lines are to be read as follows: An environment/element called `section` consists of maximum one `paragraph` and at least one `subsection` in this order. A `subsection` consists of exactly one `paragraph` plus at least one `paragraph`, e.g. at least two `paragraph`s. And at last, a `paragraph` consists of letters. Here it is not possible anymore — unlike in LaTeX — to put the first subsection before the first section. One could define the LaTeX environments with such control structures, too. But again, LaTeX is not designed for this goal and does not enforce it, while such validating is the nature of SGML.

Another structural advantage over LaTeX is the consequent distinction between *parameter* and *data*. The lines

```
\label{Hallo!}
\section{Errors}
\unknown{whatever}
```

show that in LaTeX one can never be sure what is human readable text (data) and what is internal technical information (parameter). On the other hand SGML has a strict idea of this distinction. As long as the SGML structures are not misused malevolently it is possible to make this distinction without even understanding the content. This is an important condition for any computer based data processing. An example will be given later.

But even in the days of total computerizing the final goal of text representing is to print the text onto paper. There are two projects/tools specially designed for the printing of SGML documents. FOSI (Formatted Output Specification Instance) and DSSSL (Document Style Semantics and Specification Language). But why not use LaTeX?

*Reprint from the Annals of the UK TeX Users Group **Baskerville**, Volume 5.2, March 1995. Published with permission of both Baskerville editor and author. Presented at the UK TeX Users Group conference 'Portable Documents: Acrobat, SGML, and TeX', on 19 January 1995, London, England.

LaTeX has some characteristics which make it the first choice.

- The structure of SGML and LaTeX are very close, so that the documents are easily to convert.
- LaTeX is a programming language and therefore can realize a wide range of unforeseen layouts.
- LaTeX has been used for many years by a large number of people. So there exists a widespread experience.

A principal scheme of the processing might look as shown in Figure 1.

Unfortunately it is not sufficient to convert the elements into environments and to write the needed style files. As already mentioned SGML and LaTeX have different ideas of what is data and parameters. So it is especially necessary to transform SGML-data to LaTeX-parameter so that LaTeX can handle it more flexibly. A typical example is the following:

```
<section label="main-section">
<title>section title</title>
section content
</section>
```

What one would like to get is something like this:

```
\section{section title}\label{main-section}
section content
```

One should note that `main-section` is a parameter before as well as after conversion while `section title` moves from being data to being a parameter. The easiest way to solve this problem is to introduce additional braces within the LaTeX environment. Depending on the number of parameters defined in the definition of the environment the data is treated as a parameter or the last parameter is treated as data:

```
<name parameter="value">data</name>
```

converts to

```
\Bsgml{name}{value}{%
data%
}\Esgml{name}
```

With some (yet still to be defined) command

```
\NewSgmlEnv{name}[n]{...}
```

one gets:

- both `value` and `data` being a parameter for $n = 2$.
- `value` being a parameter and `data` being data within the environment for $n = 1$.
- both `value` and `data` being data within the environment for $n = 0$.

Note that this conversion can be done without any conversion parameters. All programming, e.g. replacements are done in LaTeX. This is a major difference to the widely used SGML-to-whatever converter `format` which works with replacement tables.

But the real reason for why I started to develop my own SGML to LaTeX converter is that I felt the necessity to manipulate the data within the conversion process.

The main questions are what information about the used words are needed for typesetting and where this information comes from. Again this seems to be a typical non-English problem. In German there are two similar problems: hyphenation and (wrong) ligatures.

Basically German hyphenation rules are easily to be adapted for pattern matching and ligatures can be applied. (Hyphenation is allowed before the last consonant out of a group of consonants. There is no hyphenation within a group of consonants at the very beginning or end of a word. Certain combinations of consonants count as one single consonant. Easy, isn't it?) At the present there is a problem with the umlauts. But this problem should disappear with the `dc`–fonts. The real problem raises with complex words, e.g. words which are composed of several words but look like one. These words have to be hyphenated between the elements of the compound. This fools every pattern matching. Moreover, there should not be any ligature in these places. The reason is that one does not want to have less space 'between words'.

An example of a rather unsuspicious word is `aufflammen`. One would guess the hypenation `auff\-lam\-men`, which is wrong, of course. The english translation gives a hint: *flame up*. Within terms of `german.sty` one should write `auf"|flam\-men`, where `"|` means: hyphenation is allowed but no ligature is allowed. The printing result is 'aufflammen' instead of 'aufflammen'.

Unfortunately TeX is unable to store this information neither in the hyphenation table nor in the document preamble by `\hyphenation`. Maybe a successor of TeX will be able to do so. So far an author writing in LaTeX has to input this information directly into the document, well — if he cares...

Using a conversion from SGML to LaTeX the converter would be the right place to insert the additional hyphenation and ligature information. The converter has to use two dictionaries — a standard dictionary and a special dictionary. It is not unusual that special matters need special terms and consequently special dictionaries. But in German the problem is that one can create new complex words ad hoc. These new compounds may be specific to a particular document. So it would be a nice idea to ship this special dictionary as a structural part of the document!

In this way the author does not have to care about every single hyphenation and ligature exception, but additionally has a spell checker.

But unfortunately there is even a worse case which needs special treatment. It is the word `Baumast`, which can be `Bau\-mast` (*mast used in building*) `Baum\-ast` (*bough of a tree*), both made of wood, of course. This is a really rare case that a word must be tagged with an additional information where it occurs within the document. This information should explain which word is to be meant. One could do that in the form of an explicit hyphenation information. In SGML it could look like

```
<word which="Baum\-ast">Baumast</word>
```

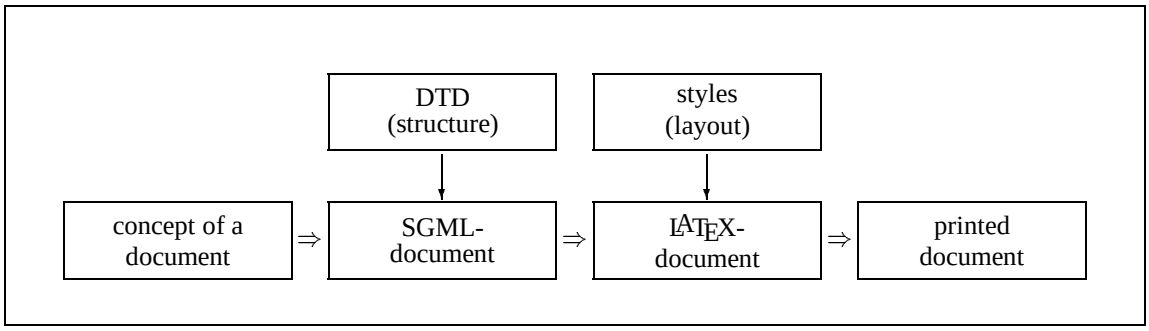


Figure 1: *Processing of a SGML document*

(This example is simplified. It would be more correct to use a SDATA-Entity so that the LaTeX-specifics are hidden.)

Note that the hyphenation information on the words `auf-flammen` and `Baumast` are totally different things. The first one is part of the layout information (how to print out?), while the second one is a structural part of the document (which word?).

Summarizing one can state that SGML and LaTeX are a good pair. Using the specifics of both systems one can do a lot of things correctly in an easier way.

Further reading

- H. Szillat: *SGML — Eine praktische Einführung* ISBN 3-929821-75-3, Int. Thomson Publ.
- ftp-server: `ftp.ifi.uio.no`
- news groups: `comp.text.sgml`, `sgml-l`

HTML & TEX: Making them sweat*

Peter Flynn

Computer center, University College,
Cork, England
cbts8001@iruccvax.ucc.ie

Abstract

HTML is often criticised for its presentation-oriented conception. But it does contain sufficient structural information for many everyday purposes and this has led to its development into a more stable form. Future platforms for the World Wide Web may support other applications of SGML, and the present climate of popularity of the Web is a suitable opportunity for consolidation of the more stable features. TEX is pre-eminently stable and provides an ideal companion for the process of translating HTML into print.

1 Markup

HTML, a HyperText Markup Language[1], is the language used to structure text files for use in the World Wide Web, an Internet-based hypertext and multimedia distributed information system. HTML is an application of SGML, the Standard Generalized Markup Language, ISO 8879[3]. Contrary to popular belief, neither SGML nor HTML is new: SGML gained International Standard status in 1986 and HTML has been in use since 1989.

SGML is a specification for writing descriptions of text structure. In itself SGML does not *do* anything, any more than, say, Kernighan and Ritchie's specification of the C language[4] *does* anything: users and implementors have to do something *with* it. It has been slow to achieve popularity, partly because writing effective Document Type Descriptions (DTDs) is a non-trivial task, and partly because software to make full use of its facilities has traditionally been expensive. It was therefore seen as a 'big business only' solution to text-handling problems until the popularisation of HTML owing to increased use of the World Wide Web. Since 1992 the software position has also improved considerably — an extensive list of tools is maintained by Steve Pfeffer at UIO[6].

2 The World Wide Web

WWW (W3 or just 'the Web') is a client-server application on the Internet. Users' clients ('browsers') request files from servers run by information providers and display them, using the HTML markup embedded in the text to render the formatting. Some of the markup can provide filenames for the retrieval of graphics as illustrations, or act as anchor-points for links to other documents, which can be further text, or graphics, sound or motion video. This latter capability gives the Web a hypertext and multimedia dimension, and allows crosslinking of files almost anywhere on the Internet.

Because the HTML files are plain text with embedded plain text markup, in traditional SGML manner, they are immediately portable between arbitrary makes and models of computer or operating system, making the Web one of the first genuinely portable, multiplatform applications of its kind.

2.1 HTML Markup

An example of simple markup and an appropriate rendering is illustrated in Figure 2. The conventions of SGML's Reference Concrete Syntax[3] are used, so markup 'tags' are enclosed in angle brackets (less-than and greater-than signs), in pairs surrounding the text to which they refer, with the end-tag being preceded by a slash or solidus immediately after its opening angle bracket.

The rendering is left almost entirely to the user's client program, as there are almost no facilities within HTML for the expression of appearance apart from a minimal indication of font change (italics, boldface and typewriter-type). Indeed, most recent browsers allow the *user* arbitrary control over which fonts, sizes and colours should be used to instantiate the tagged elements of text.

2.2 Implementation

HTML was devised for the Web by non-SGML-experts who saw it as an ideal mechanism for implementing plain-text portability while preserving sufficient structural information for online rendering: one of the classical reasons for adopting SGML. It is now becoming standardised by an IETF working group who have produced a draft specification in the form of a formal DTD[1]. Because of the need to allow this specification to model existing 'legacy' documents (most of which would be regarded as fragments

*Reprint from the Annals of the UK TEX Users Group **Baskerville**, Volume 5.2, March 1995. Published with permission of both Baskerville editor and author. Presented at the UK TEX Users Group conference 'Portable Documents: Acrobat, SGML, and TEX', on 19 January 1995, London, England.

```
#! /bin/sh

echo Content-type: text/html
echo

cat <<EOH
<html><head><title>Date and time</title></head><body><p>It is now
EOH
date
cat <<EOT
</p></body></html>
EOT
```

Figure 1: *Example of a Unix shell script to return the date and time as a HTML file*

```
<html>
  <head>
    <title>Fleet Street Eats</title>
  </head>
  <body>
  <h1>Where to eat in Fleet Street</h1>
  <p>There are many restaurants in the City, from
     fast-food joints to <i>haute cuisine</i>.</p>
  ...
```

Document title: `Fleet Street Eats`

Where to eat in Fleet Street

There are many restaurants in the City, from fast-food joints to *haute cuisine.*

. . .

Figure 2: *Example of HTML markup and possible rendering*

rather than document instances), as well as provide for more robust usage, the current DTD has two modes: a non-rigorous 'deprecated' mode for describing the legacy and a 'recommended' mode for creating and maintaining files in conventional form.

HTML is sufficient for minimal documents, providing the structural and visual features shown in Figure 3. A future version (3.0) is being developed by the IETF Working Group, which will allow the description of mathematics, tables and some additional visual- and content-oriented features.

Despite the coming improvements, HTML is likely to be joined in the Web by other DTDs in future. One well-known SGML software house already has a prototype browser which can handle instances of arbitrary DTDs, given sufficient formatting information. This would make it possible to use the Web for transmission and display of documents using other SGML applications such as CALS (US Military), DocBook (O'Reilly/Davenport), the TEI (Text Encoding Initiative) and corporation-specific DTDs (such as those of Elsevier).

The next version of the DTD, HTML3, contains specifications for mathematics, tables and some additional elements for content-descriptive material, as well as a few extra visual keys such as an `ALIGN` attribute for positional specification. Most of this work is being implemented on a test basis in the Arena browser (Unix/X only at the moment) at CERN.

Although Web browsers can reference files by any of several methods (HTTP, the Web's 'native' protocol; FTP; Telnet; Gopher; WAIS; and others) by using the URL (Universal Resource Locator: a form of file address on the Internet), the most powerful tool lies at the server end: the ability of servers to execute scripts, provided their output is HTML. A trivial example is shown in Figure 1, which returns the date and time.

Such a script can contain arbitrary processing, including the invocation of command-line programs and the passing of arguments. Data can be gathered from the user either with the `<isindex>` tag in the header, which causes a single-line data-entry field to appear, or with the more complex `<form>` element with scrollable text boxes, checkboxes, radio buttons and menus. In this manner, complete front-ends can be manufactured to drive data-retrieval engines of any kind, provided that they operate from the command line, and that the script returns their output in HTML. The user (and the browser) remain unaware that the result has been generated dynamically.

Structural		Descriptive		Visual	
`html`	document type	`a`	hypertext link anchor-point	`b`	bold type
`head`	document header	`cite`	citations	`br`	forced line-break
`title`	document title	`code`	computer code	`hr`	horizontal rule
`base`	root address for incomplete hypertext references	`em`	emphasis	`i`	italics
`meta`	specification of mapped headers	`kbd`	keyboard input	`tt`	typewriter type
`link`	relationship of document to outside world	`samp`	sample of input	`img`	illustrations
`isindex`	specifies a processable document which can take an argument	`strong`	strong emphasis		
		`var`	program variable		
`body`	contains all the text				
`h1...h6`	six levels of section heading				
`p`	paragraph				
`pre`	preformatted text				
`blockquote`	block quotations	**Form-fill**		**Obsolete:**	
`address`	addresses	`form`	contains a form	`listing`	use `pre`
`ol`	ordered lists	`textarea`	free-text entry	`xmp`	use `pre`
`ul`	unordered lists	`input`	input field (text, checkbox, radio button, *etc*	`plaintext`	use `pre`
`menu`	menu lists	`select`	drop-down menu	`nextid`	editing control
`dir`	directory lists	`option`	menu item	`dfn`	definition of term
`li`	list item				
`dl`	definition lists				
`dt`	definition list term				
`dd`	definition list description				

Figure 3: *Markup available in HTML 2.0 (indentation implies the item must occur within the domain of its [non-indented] parent)*

2.3 Presentation

HTML is criticised for being 'presentation-oriented', but as can be seen from Figure 3, the overwhelming majority of the markup is structural or content-descriptive. However, this does not prevent the naïve or sophisticated author from using or abusing the markup in attempts to coerce browsers into displaying a specific visual instantiation, primarily because none of the browsers (with the partial exceptions of Arena and `w3-mode` for GNU Emacs) performs any form of validation parsing, and will thus display any random assemblage of tags masquerading as HTML. This behaviour has misled even some eminent authorities to dismiss HTML as 'not being SGML'.

There is thus a conflict between the SGML purist on the one side, who decries any attempt at encoding visual appearance; and the uninformed author on the other, who has been unintentionally misled into thinking that HTML and the Web constitute some kind of glorified networked DTP system.

The purists are few in number but eloquently vocal: however, in general, they acknowledge that visual keys can be included if they are carefully coded. A perceived requirement to allow an author to recommend the centering of an element is thus achieved in HTML3 by the `align="center"` attribute, rather than the unnecessary `<center>` element proposed by the authors of Netscape.

The demands of the author are at their most marked in the approach of publishers and marketing users, who have been accustomed for the last 550 years to exert absolute control over the final appearance of their text. But the Web is not paper, and the freedoms and constraints of the Press do not apply: it is as much a new medium as radio or television. For such an author to insist that she must be able to control the final display to the same extent as on paper is as pointless as insisting that a viewer with a black-and-white television must be able to see the colours in a commercial.

The paradigm has been established that the browser controls the appearance, using the markup as guidelines. There is indeed no reason at all why attributes could not be added so that an author could write

```
<h1 color=green font=LucidaBrightBoldItalic
                              size=24 shading=50>
```

but the user of Lynx or WWW (two popular text-only browsers for terminal screens) would still only see the heading in fixed-width typewriter characters. The habit of insisting that everyone 'must' see a particular typographic instantiation is an unfortunate result of a misinterpretation of the objective of the Web: to deliver information in a compact, portable and arbitrarily reprocessable form.

But publishers accustomed to paper, insistent on 'keeping control', have of course an entirely valid point, one with which the present author has great sympathy. Why should a carefully-prepared document be made a hames of by a typographically illiterate user who has set `<h2>` to display as 44pt Punk Bold in diagonal purple and green stripes?

The solution probably lies in the implementation of style sheets, perhaps along the lines of those discussed by the authors of Arena[5]. They would in any case only be recommendations: not every user has a CD-ROM of Adobe or Monotype fonts. In any event, if 100% control is essential, as in the display of typographic examples, all graphical browsers can be configured to spawn a window to display PostScript file, although the download time may be a strong disincentive.

It is entirely possible that the control of content will ultimately prove a more attractive option than the control of appearance.

3 Publishing with HTML

Setting aside the unresolved questions of display, there are more pressing business problems about publishing on the Web.

The authentication of users is being addressed at several levels, from simple, non-authoritative checks using `identd` to the more complex username-and-password systems employed on some Web pages. From the user's end, the authentication of the data being accessed is equally important. The openness of the Internet in its raw form allows 'spoofing' in both directions, so the emergence of protocols to provide checks is to be welcomed.

The security of network-accessible texts from break-ins remains a concern to anyone providing high-value merchandise, and Web text is in this sense no different from any other computer data. Normal precautions must therefore be taken to prevent theft through other channels (such as remote login), as distinct from theft perpetrated by falsification of Web access.

There is a need for robust solutions to charging and billing for usage, and the secure transmission of financial data, including credit card numbers, digital signatures, and perhaps even EFT transactions. The Secure HTTP (SHTTP) mechanism being marketed by MCom and others is becoming popular as a way of achieving some of this, but the Internet must shed some of its image of lax controls and sloppy housekeeping if it is to achieve sufficient 'respectability' to attract the business of those who are not networking specialists.

The handling of copyright and the intellectual property of electronic texts remains, as ever, an unsolved problem. While copyright law can be used to provide a remedy for breach, the difficulty lies in preventing the breach occurring in the first place. The reason is that (as with other electronic material), copying and reproduction is fast, cheap and easy, once the material is in the hands of the customer. While a supplier may use SHTTP to protect the details of the transaction, once a print file has been sent to someone, the supplier retains no control whatsoever over its use, reuse and abuse. Copies could be sent to dozens others, or printed many times, in the space of minutes.

3.1 Printing from HTML

The demand for printed copies of Web material is surprisingly high. Although in some cases it is reminiscent of those people who insist on printing their email, it is undeniable that there is a serious requirement for good quality print from Web documents.

Existing solutions to printing SGML text are usually application-specific, being embedded in SGML editors or DTP systems, but there are also some more generic packages:

- `Format` by Thomas Gordon (LaTeX)
- `HTMLtoPS` by Jan Kårrman (PostScript)
- `SGML2TeX` and `WebSet` by Peter Flynn (TeX/LaTeX)
- `SimSim` by Jonathan Fine (TeX)

The use of TeX systems for most of these seems to indicate that the similarity of markup concepts has not gone unnoticed by practitioners. The author's own contributions are experimental, but the second of them is planned as an interactive Web service, to be introduced in the summer of 1995. Emailing a URL to the point of service will cause it to be retrieved, typeset, and the output returned to the user by email in PostScript form. As a form of email browser, the control of appearance may lie in the hands of the user, but suggestions for how implement this are currently being sought[2].

3.2 Problems

Implementing a professional level of typesetting from HTML raises some interesting questions:

- most HTML files are invalid
- most HTML authors don't understand SGML
- most HTML authors couldn't care less
- most World Wide Web users couldn't care less

The handling of missing, damaged or abused tags in a gracious manner is not a feature of most SGML parsers. At the best, a typesetter-browser can only be expected to report to the user that a file is invalid, and while it may be displayed by browsers which do not make any claim to typographic quality, an attempt to make a respectable print job of an invalid file is unlikely to succeed.

4 Development

The future of the World Wide Web and HTML is uncertain. While development continues, and while new users are anxious to start surfing the net, the existing designs and implementations will suffice. In the longer term, a coalescing of services is likely to occur, but for this to happen, a number of changes need to take place:

- The Web will start to make use of other DTDs, as outlined above. Any file containing a `<!doctype...>` at the beginning could cause a browser to retrieve the DTD specified, along with a style sheet, and work much as any SGML-conformant DTP system would.
- Browsers will become pickier, able to offer better services at the expense of rejecting invalid or badly broken files. Arena already perfoms a form of consistency check on the HTML code of files, and displays 'Bad HTML' in the top corner when an offender is spotted.
- Users will become pickier, demanding better response from the browser, better response from the server, and better facilities from both. As users become more educated about the use of SGML, developers will no longer be able to hide the deficiencies of products under the cover of technical detail.
- This presupposes more user education, which is inevitable in a developing technology. 100 years ago, motor cars appeared on the roads, but few passengers in them understood the use of the levers and rods which controlled them. With some minor exceptions, it is now expected that a driver knows that turning the wheel clockwise turns the car to the right, and *vice versa*. It will not take us that long to perceive the innards of HTML, but it can only be done by training and education.
- At some stage, investment is always needed. Many companies have put substantial sums into the development of Internet resources, and those that have done so with forethought and planning deserve to reap a rich reward. It is a long-term investment, more akin to a partnership, but support is always needed by those who undertake the developments, especially as much of it is done in personal time and at personal expense.

There is still some way to go before we achieve the ease of use of the telephone or the radio, but the path is becoming easier with each new development.

References

[1] Berners-Lee T & Connolly D, *HyperText Markup Language Specification — 2.0*, Internet Draft, IETF Working Group on HTML, December 1994.

[2] Flynn P, *Typographers' Inn*, TeX and TUG NEWS, **4**, 1, March 1995.

[3] Goldfarb C, *The SGML Handbook*, OUP, 1990, ISBN 0–19–853737–9.

[4] Kernighan BW & Ritchie DM, *The C Programming Language*, Prentice-Hall, 1978.

[5] Lie H *et al*, *HTML Style sheets*, `http://www.w3.org/hypertext/WWW/Style/`

[6] Pepper S, *The Whirlwind Guide: SGML tools and vendors*, `ftp://ftp.ifi.uio.no/pub/SGML/SGML-Tools/SGML-Tools.txt`

The Inside Story of Life at Wiley with SGML, LaTeX and Acrobat*

Geeti Granger

John Wiley & Sons Ltd,
Baffins Lane, Chichester, W. Sussex PO19 IUD, England

1 Introduction

As a brief introduction I should say that John Wiley & Sons is a scientific, technical and medical publisher. It is an independent, American family-owned company that was established in 1807, with subsidiaries in Europe, Canada, Australia and Singapore. The European subsidiary opened in London in 1960 and moved to Chichester in 1967 (if folklore is to be believed this was so that the then Managing Director could more easily pursue his love of sailing!).

We publish books, including looseleaf and encyclopaedias, and journals, and most recently electronic versions of some of our printed products. In the future the electronic component of our publishing programme is bound to include products that are only available electronically.

2 Setting the Scene

Now to the topic in hand — Portable Documents: Acrobat, SGML and TeX. Our association with TeX dates back to 1984 when we made the significant decision to install an in-house system for text editing and composition. It was the only software available that wasn't proprietary, which stood a chance of coping with the complex mathematical material we had to set.

As a company we have monitored the progress of SGML since 1985, but have only recently used it in earnest. Our first project is a 5000 page encyclopaedia about Inorganic Chemistry. We rarely get the opportunity to dip our toes in the water — it's straight in at the deep-end! Having said this, we do have a set of generic codes that has been used for a number of years, and everyone is well aware of the principles involved and the value of this approach to coding data.

Adobe Acrobat was launched in June 1993. Our experience of this software dates back a little further than this, because of our links with Professor David Brailsford and the Electronic Publishing Research Group at the University of Nottingham, and their work on the CAJUN (CD-ROM Acrobat Journals Using Networks) project, which we jointly sponsored with Chapman & Hall.

3 Complementary not Competitive

The first thing to make clear is that SGML, TeX and Acrobat do not compete with each other in any way. SGML is a method of tagging data in a system-independent way. TeX is one possible way of preparing this data for presentation on paper, while Acrobat is software capable of delivering data electronically for viewing on screen, or for committing to paper.

From our point of view the fundamental requirement for:

- capturing data
- processing data (text and graphics)
- delivering data (paper/disk/CD/Internet)

is to remain system independent for as long as possible.

SGML, TeX and Acrobat achieve this in their part of the whole process. PostScript provides the link that completes the chain.

4 SGML in Practice

To describe our experience with SGML I will use the *Encyclopedia of Inorganic Chemistry* as a case study. This encyclopaedia is an 8 volume set made up of 5000 large-format, double-column pages (more than 3 million words). The data consists of approximately 250 articles interspersed with 750 definitions and 750 cross-reference entries. The text was marked-up and captured using SGML, validated and preprocessed for typesetting. The floating elements (all 2300 figures, 8000 equations, 2000 structures, 1100 schemes and 900 tables) were prepared electronically and delivered as encapsulated PostScript files. Some 150 halftones, about a third of which are colour, complete the data set!

Despite the complex nature of this project, or maybe because of it, we were convinced that using SGML was the right approach. We had to be very sure because this decision presented us with many additional difficulties. Different considerations had to be made at all stages of the production process. (Manufacturing remained untouched.)

Initially, having established the probable requirement for an electronic version, there was the need to justify the use of SGML because of:

- the extra cost involved in data capture

*Reprint from the Annals of the UK TeX Users Group **Baskerville**, Volume 5.2, March 1995. Published with permission of both Baskerville editor and author. Presented at the UK TeX Users Group conference 'Portable Documents: Acrobat, SGML, and TeX', on 19 January 1995, London, England.

- the different working practices that had to be established
- the project management overhead
- the need to find new suppliers, and the risks that this involved for such a large, high profile project.

4.1 Production Considerations

This project had an external Managing Editor to commission and receive contributions before it became a live project for us. Once contributions started to arrive it very quickly became apparent that a project management team was needed if this project was to succeed. The initial steps had to be ones of project analysis, determining data flow, deciding who was responsible for what, and ensuring that a progress reporting system was established. It certainly seemed like a military operation at times.

Having made the decision to go with SGML and to ensure that all components were captured electronically we had to find a set of new suppliers. None of our regular suppliers could meet our specifications. Locating potential suppliers was the first hurdle, and then assessing their suitability was the next. Having done this we then had to draw them all together to establish who did what, and who was responsible for what. It had to be a team effort from start to finish and regular progress meetings involving representatives of all parties was the key to an ultimately successful project.

4.2 Problems Encountered

One of the first considerations was how on earth do we name the files? To ensure portability we set ourselves the restriction of the eight plus three DOS convention. It took some time but we achieved it in the end so you can now identify from the file name the type of text entry, the type of graphics and whether it is single or double column or landscape and its sequential placement within its type. When you consider the number of files involved, this was no mean feat.

Designing the DTD without all the material available is not the best way to start, but needs must. It meant that some amendments had to be made as the project progressed but none of them proved to be too significant.

Choosing Adobe typefaces, to avoid problems later on, meant that some compromises had to be made. Many people feel that the Adobe version of Times is not as elegant as some.

Also the quality of the typesetting, hyphenation and justification, interword spacing and overall page make-up is not as high as that normally achieved by a dedicated chemistry typesetter.

In addition to the above, we found a bug in Adobe Illustrator! Because the EPS files were being incorporated electronically the accuracy of the bounding-box coordinates was crucial. To cut a long story short they weren't accurate. We spent quite some time establishing the cause of the problem and then had to have a program written to resolve it.

This is not an exhaustive list but I think it will give you a feel for the practical issues involved. Having shared all this with you I should add that all of us involved in the original recommendations remain convinced that it was the right approach. In fact we are now processing two more projects in the same way!

5 LaTeX in Practice

We've done far too many projects in TeX (many in Plain, but a growing number in LaTeX) to select one as a case study. What I can do is very readily identify the production issues involved in using this software in a commercial environment.

5.1 Steps in the Process

Establishing ourselves as a forward-thinking, progressive company by developing in-house expertise has brought with it certain pressures. In the early days, not only did we have to learn how to use TeX, we also had to make it achieve typesetting standards expected of more sophisticated systems. Our colleagues could not see why they should accept lower standards from us — after all they were paying us (we operate a recharge system so that it doesn't distort the project costing when compared with externally processed projects).

Next came the requests for us to supply style files. Authors knew we used the same software as they did, and wanted to prepare their submission so it looked like the finished product. Some wanted to produce camera-ready copy. In principle this would seem a sensible idea; in fact our commissioning editors, especially those who handle a number of CRC projects, thought it was a brilliant idea. It would save them an immense amount of time and hassle.

Now, preparing style files for in-house use is one thing; preparing them for use by others is something else again. We have to work within strict time and cost constraints, and there are many occasions (dare I admit it?) when we have to resort to, shall we say, less than the most sophisticated way of achieving the required visual result!

When I have attended courses on TeX and have asked about writing style files the answer has often been along the lines of 'leave it to the professionals'. (I should say it's usually people who make their living in this way who give this response.) This may be fine if a) you can find and afford the professional; b) you don't need to support the file when it is in general use. In our experience the first is difficult to do and the second is an impossibility. The need to support style files cannot be ignored; once they have been provided, no matter on what pre-agreed conditions, queries will arise. It can be very time-consuming, as often queries are not restricted to the style file, but relate to the sytem being used. It can also take a while to establish the context of the query, resolve it and respond. To meet the expectation that we will support, customise at short notice, resolve technical issues, and communicate via e-mail (preferably responding within the hour) can be difficult, given the level of human resource available.

Once you've got over this initial stage, the practical issues involved in accepting LaTeX submissions can be many. Delivery is the first. Now that we have the ability to receive data electronically our authors cannot understand why we hesitate, and why we still insist on hard copy. Experience tells us that, without hard copy, it is difficult to be sure we have received the final version, and discovering this after a project has been processed is very costly, both in time and money. Any submission that circumvents a stage in the current administration process may drop through a hole and end up taking more time, rather than less, to reach publication. Consideration is being given to this issue, and there is no doubt that in the future electronic delivery will be an acceptable method of submission, but in the meantime everyone has to be patient.

Copy-editing remains a conventional process in the main, although experiments are taking place with copy-editing on disk. This issue is not resticted to LaTeX projects, but the rate of progress is dictated by the ability of our freelance copy-editors to provide this service.

Once you move on to the processing stage the first thing you have to do is find a supplier who is capable of actually processing in this software. This is easier said than done, because it is not considered to be cost-effective by most of our regular suppliers. However, as a result of our persistent requests, some can now provide this service, so we don't have to process all such submissions in-house.

From our own experience we know that producing page proofs is not always straightforward. Over the years we have struggled with amending style files to achieve the correct layout and controlling page make-up. Now that authors are submitting graphics on disk, as well as the text, we are faced with another set of problems. Portability of graphic formats is even more difficult to achieve. I think the number of answers to the question 'When is a PostScript file (or EPS file) not a portable PostScript file?' must be infinite. Even when the content of the file itself is OK, you can still be faced with problems in achieving the required size and position on the page.

Despite all these disadvantages our lives would not be the same without LaTeX, and when compared with processing in other software it can be a real joy! Our archive of projects coded in a form of TeX will be far easier to reuse than those processed in other software.

6 Acrobat at Arm's Length

Although we haven't used Acrobat on a live project in-house yet, we have been closely involved with the development of the EPodd CD. The CAJUN project has been running for well over a year and during this time the complete archive of volumes 1–6 has been converted to PDF, annotated to add PDFmarks and generally massaged into a suitable format for delivery on CD.

As always, the work involved in such a project is more than anticipated at the outset, but it has been an invaluable learning exercise. Being involved in the beta-testing of the software helps you appreciate just how much development work is required for a new piece of software, and although it currently has its limitations the future looks good. Version 2, which is due for release any day now, is much improved, and it is rewarding to see that many of the comments put forward by members of the team have been incorporated.

We are experimenting with small projects in-house to give us a deeper understanding of the practical advantages and limitations of Acrobat. It is easy to get caught up in the euphoria and hype that accompanies the release of a new product, and to overlook the day-to-day difficulties its rapid adoption might bring. Having said this, there is no doubt that it will have a place in our publishing procedures, and may be used in the production cycle for journal articles. Provided that the general administration can cope with the deviation from the norm, supplying author proofs in this way has its attractions. The fact that readers are now freely available and the PDF file can be read on any of the three main platforms is a real boon.

The use of Acrobat for delivering existing print products in an electronic form is one worth considering, especially now that it is possible to integrate it with project-specific software and the security issue has been addressed.

From an inter-company point of view the perceived use of Acrobat for distributing internal documents could again have its attractions. For this to be a real possibility it must be recognised that the use of such procedures is not an innate skill, and so the appropriate level of training and support must be available if it is to be successful.

7 Conclusion

The comments I have made and the case study I have described may leave you with a somewhat negative feeling. I wonder if I have emphasised the problems and not balanced these by identifying the plus points. To put this into context I should say that details of the advantages of any particular approach are usually more readily available, so I have tried to capture a more down-to-earth view.

In reality I am very enthusiastic about the use of SGML, TeX and Acrobat, but am also well aware of what their use in a productive environment can mean. I believe, as do several of my colleagues, that portability of documents is crucial to our ability to deliver data efficiently in a variety of forms, whether this be page-based, highly structured databases or tagged ASCII files. To this end we must be flexible in our approach, and must not be afraid of making investments now that may not bear fruit until some time in the future. This can be a very unnerving decision to make, and for one I am glad it isn't ultimately mine. While I can extol the virtues of a purist's technical approach, obtain the relevant costs and assess the schedule implications, I do not have the entrepreneurial skills required to know when a project is commercially viable (or worth taking a risk on). It is at this point I take my hat off to our commissioning editors, who have the responsibility for turning these experiments into profit for us to reinvest in the next Big Thing!

Theory into Practice: working with SGML, PDF and LaTeX at Elsevier Science*

Martin Key

Elsevier Science Ltd, England
m.key@elsevier.co.uk

1 The Company

While I do not want to make this article a plug for Elsevier, it is first necessary to put our activities into context. Therefore, for those who do not know us, Elsevier Science is part of the Reed Elsevier Group and, in terms of number of journals, is by far the largest publisher of scientific journals in the world. The original Elsevier Company was Dutch based, but now, through acquisition and merger, is an international company with offices in the Netherlands, UK, USA, Switzerland, Eire and the Far East. We publish well over 1,000 scientific, technical and medical journals covering all sections of academe and business.

2 The move into electronic publishing

Elsevier's major customers are academic and research institutes throughout the world. Traditionally, academic publishing has relied on authors submitting papers via external academic editors who arrange for the necessary peer reviews. Once accepted, papers are sent to Elsevier for copy-editing, typesetting and compilation into issues. As a result we have in the past received paper manuscripts of varying levels of presentation from around the world. Over the last 10 years it has become apparent that most authors use some form of word processing or computer generated text to prepare their papers. To have these papers typeset means rekeying the manuscript and, what is worse, ending up with electronic files produced by many types of typesetting equipment and software with minimal chance to reuse this material at a later date. For some years the Elsevier Group have been looking at ways to avoid rekeying manuscripts whilst at the same time automating the production process, produce proofs more quickly and create electronic files for multiple use in the foreseeable future.

After many surveys, experiments and discussion groups it was clear that Elsevier should work to accepted international generic standards in order to achieve these goals. The major standards agreed on were Standard Generalised Mark-up Language (SGML) for text, Tagged Image File Format (TIFF), Joint Photographic Experts Group (JPEG) and Encapsulated PostScript (EPS) for graphics and PostScript, and the Portable Document Format, (PDF), also known as Acrobat, for pages. Unlike typesetting codes, SGML does not drive any particular application but can be readily converted to numerous formats for typesetting on paper, database applications, CD-ROM and so on. It is therefore an ideal archive medium. TIFF, JPEG and EPS are well documented graphic file formats and are widely supported in terms of external applications. PDF is, perhaps, a risk in that it is the property of a commercial developer (Adobe) but its great flexibility and rapid acceptance by professionals and the academic community, together with the track record of PostScript itself — now a de facto standard — makes its long-term future seem relatively safe. The decision by Adobe to make the Acrobat reader available free-of-charge is another positive sign.

3 The concept of Computer Aided Publishing (CAP)

Once the standards were agreed the process known internally as CAP (Computer Aided Publishing) took clearer shape. There are a number of activities which form part of CAP. These include the following: the converting of manuscripts and artwork into electronic files; structuring of text with SGML; editing on screen; automatic proofing; moving and maintaining files on a network; creating SGML (text) and graphic files; receiving PDF files from our typesetters. In addition, a number of journals receive, and use, papers in LaTeX format which will be discussed later.

4 Practicalities: How we do it

CAP started in Elsevier in January 1994, in both Amsterdam and Oxford, with a limited set of journals. The number of journals has been increasing rapidly and in 1995, as software and hardware stabilises, the number of journals is being increased dramatically.

The first action when receiving a paper, either on paper or disk, is to log the information on to our production tracking system. All the important details are recorded — title, authors, number and type of graphics, whether it is available on disk etc. This record follows the manuscript throughout its production process and is updated at each stage of its progress through the system. Elsevier encourage authors to submit on disk, and the numbers are rising. If it is on disk it is initially converted to our standard CAP format which

*Reprint from the Annals of the UK TeX Users Group **Baskerville**, Volume 5.2, March 1995. Published with permission of both Baskerville editor and author. Presented at the UK TeX Users Group conference 'Portable Documents: Acrobat, SGML, and TeX', on 19 January 1995, London, England.

allows it to be used by our SGML tagging and editing tool — Pandora — which was developed by staff working in Amsterdam. If it is only available on paper it is either OCR (Optical Character Recognition) scanned and then converted into the CAP format or, if the paper is too complex for scanning, it is keyed by off-shore keying agencies. Whatever the route, it arrives at our Pre-Edit Department in the generic CAP format. Simultaneously graphics are scanned — TIFF for line art and JPEG for half-tones — or redrawn and saved as EPS in some instances.

The text is then tagged using Pandora. The Document Type Definition (DTD) used is the Elsevier DTD (which Elsevier has made publicly available subject to certain conditions) which is fairly complex covering not only text but also tables and mathematics.

After coding and parsing, the text is loaded onto the network server, together with the graphics, using an in-house developed Document Management System which monitors, names and controls the files. As one article can produce more than 20 files, with an average issue of a journal containing 10 articles, the number of files can quickly mount making such management essential. Once the files are on the server, they can be retrieved by the Production Editor who will then edit the article for style, spelling, grammar, etcetera and add any additional tags necessary. Graphic files are also checked at this stage to ensure that the correct graphics are linked to the relevant caption. The file is then parsed again to check its validity. Author proofs can then be produced and, once they are received back from the authors and corrections made, the final SGML and graphic files are exported to the typesetter for making up the final pages.

We expect typesetters to retain the validity of the SGML files when producing the pages, and this is strictly monitored. Due to the complexity of the DTD and the relevant inexperience of most typesetters in using precoded SGML files, we have to work with our typesetters quite closely, answering specific queries and offering advice where necessary. However, we do not expect to develop the systems for the typesetters — that is their responsibility. The final, additional requirement we demand from our typesetters is that they supply each individual article, and other elements of the issue, in PDF format. This means that they must have a PostScript setter in order to create these files.

5 TeX and LaTeX

In some disciplines TeX and LaTeX are used extensively by authors and, not unnaturally, they would like to submit their articles in this format. Experience has shown that this can be hard work for the Publisher. In some cases, hacking in to such a file to find out how the author's carefully developed macros have been used can be very time-consuming and, in some cases, can take considerably longer than having the paper professionally typeset. However, whenever possible, we will try and use submitted LaTeX files and, to a lesser extent, plain TeX files. However, Elsevier encourage authors to use the Elsevier style file which produce a pre-print type output. This style is then replaced with the journal-specific style file which makes the Publisher's task considerably easier. The Elsevier style files, together with the instruction manual, are available from the three CTAN sites or direct from Elsevier.

LaTeX has a number of advantages. Pages in camera ready format can be produced readily in-house without recourse to a typesetter, and PDF files can also be generated from the dvi files. Recently, the Production Methods Group at Elsevier Science Ltd has further developed the 'dvihps' converter and LaTeX macros from the HyperTeX project, to fully retain the hypertext links available in the LaTeX file, as well as generating automatic 'bookmarks' or contents list, directly into the PDF file. In order to meet the full CAP requirements previously mentioned, there is one final part of the equation to be completed — a LaTeX to SGML conversion. Due to the complexity of the Elsevier DTD this is not a simple task but work is currently taking place to see how far down this road it is possible to go.

6 Practical Problems

As with most technical developments there are always problems to be addressed. In the case of CAP they have been surprisingly few. The major problem experienced at an early stage was the lack of SGML editors which could cope with the Elsevier DTD, particularly in the area of tables and mathematics. This problem has been largely resolved by the development of Pandora, a tool which has far exceeded its initial specification as a package which would enable compuscripts to be handled by typesetters. The second problem was one of logistics — how do you train Production Editors to work with SGML on screen editing whilst simultaneously producing journal issues? As previously mentioned, there is also the increased demand we place on typesetters, many of whom have had limited experience of handling complete journals in SGML. Finally, as Production Editors began to use the DTD in earnest, additional requirements are discovered which means that the DTD must be further developed. As a result, the DTD has become a moving target with more complex requirements being asked for almost daily.

7 The Future

Some people may ask why we are putting ourselves through so much pain. Is it worth it? The market is demanding electronic products in addition to, and sometimes instead of, the traditional paper ones. For those publishers who have tried to use typesetters' tapes for such products, the answer is clear. The availability of generic coded data which can be manipulated in multifarious ways is clearly the route to take. In addition to meeting the demands of our market, we are also satisfying the demands of our producers — the authors — who create 'electronic' versions of their articles and who naturally expect that we, the Publishers, should be able to use them. Finally, the Production process itself is being streamlined allowing for more efficient and faster production times.

LaTeX2HTML Update '95

Nikos Drakos
MatriX Publishing Network,
32–34 Broadwick Street, London W1A 2HG
nikos@mpn.com
http://www.mpn.com/

1 Introduction

A new version of LaTeX2HTML (version 95.1) has been available since January 1995. This article outlines some of the main changes.

2 Brief overview of LaTeX2HTML

LaTeX2HTML is a conversion tool that allows documents written in LaTeX to be converted into the hypertext format (HTML) used by WorldWide Web navigators.

LaTeX2HTML recreates the basic structure of a paper document as a set of interconnected hypertext nodes which can be explored using automatically generated navigation panels. Any cross-references, citations, footnotes, etc are converted into hypertext links. Formatting information (eg for mathematical equations or pictures) is converted into images which are placed automatically in the hypertext document.

LaTeX2HTML is being widely used for the preparation of active electronic books, online documentation, electronic scientific papers, lecture notes, training and coursework material, literate programming tools, active bibliographic references and much more. Several thousands of copies of LaTeX2HTML have been distributed to academic, commercial and government institutions since May 93 when it was first released.

3 LaTeX2HTML 95.1 new features summary

- **Much improved inlined equation baseline alignment!**
  (Thanks to Mark Segal, segal@spud.asd.sgi.com) Inlined equation bitmaps are now aligned correctly depending on whether they contain subscripts, superscripts etc.
- **Support for internationalization**
  (Thanks to Martin Boyer, gamin@ireq-robot.hydro.qc.ca) A global variable LANGUAGE_TITLES can now be used to change the language in which some section titles (eg 'Table of Contents') are printed. It is also very easy to add support for more languages.
- **More efficient implementation**
  There has been a major overhaul of the way the source text is parsed and analysed in order to reduce the memory requirements of this process.
- **'Off-line' Image Generation**
  Two new options -no_images and -images_only allow 'off-line' image conversion. The advantage of using these options is that the translation can be allowed to finish even when there are problems with image conversion. In addition it may be possible to fix manually any image conversion problems and then run LaTeX2HTML again just to integrate the new images without having to translate the rest of the text.
  Also, a new option map=<image map URL> in the command htmlimage can turn an included postscript image into an active image map.
- **Compatibility with Perl 5**

4 Examples of converted documents

Electronic books

- Spinning the Web,[1] a book for WWW developers by Andrew Ford.
- Designing and Building Parallel Programs (Online)[2] which is an 'evolving online resource' incorporating the content of a 500-page textbook published by Addison-Wesley.
- Common Lisp the Language, 2nd Edition[3] — an electronic version of the 1000+ page *LISP bible* by Guy L. Steele, published with permission from Digital Press.
- Computational Science Education Project.[4]

Scientific papers

- The MIT transit project,[5] and a paper on electronic submissions to an IEEE journal.[6] Also, some sam-

[1] http://power.globalnews.com:80/stw/stw/home.htm
[2] http://www.mcs.anl.gov/dbpp
[3] http://www.cs.cmu.edu:8001/Web/Groups/AI/html/cltl/cltl2.html
[4] http://csep1.phy.ornl.gov/csep.html
[5] http://www.ai.mit.edu/projects/transit/tn-cat.html
[6] http://www.research.att.com/esubmit/esubmit.html

ple journal articles in SEPTEMBER (AT&T's Secure Electronic Publishing Trial[7])

Training and teaching support material

- ISLE — The Intensely Supportive Learning Environment[8] project at ICBL, Heriot-Watt University, lecture notes at Cardiff,[9] and at Brigham Young[10] Universtities.

System documentation

- The REDUCE algebra system,[11] the PYTHON tutorials[12] and the user guide to the Compton Observatory Science Support Center.[13]

5 Where to get it, how to use it

LaTeX2HTML runs on Unix systems (SunOS, OSF, Linux, AIX etc) with at least Perl version 4 at patch level 36. Versions of LaTeX2HTML earlier than 95.1 will *not* work with the newer Perl version 5.

You can get LaTeX2HTML from the 'Getting LaTeX2HTML' section of the LaTeX2HTML manual or via ftp from

`ftp://ftp.tex.ac.uk/pub/archive/support/latex2html`

or other CTAN archives.

Fully searchable archives of user comments, bug reports, the LaTeX2HTML user manual, the LaTeX2HTML mailing list, links to converted documents, the source code, and much more are available from the LaTeX2HTML server.[14]

6 Appendix (added by MAPS editor)

6.1 The LaTeX2HTML Mailing List

A LaTeX2HTML mailing list has been set up. To join send a message to:

`latex2html-request@mcs.anl.gov`

with the contents

`subscribe`

To be removed from the list send a message to:

`latex2html-request@mcs.anl.gov`

with the contents

`unsubscribe`

Please send these messages to `latex2html-request` and *not to everybody else on the list.*

The address of the LaTeX2HTML mailing list is:

`latex2html@mcs.anl.gov`

The mailing list archive is available.

6.2 Future support and further development

On 3 Apr 1995, the following e-mail was distributed on the `latex2html@mcs.anl.gov` discussion list by the LaTeX2HTML author Nikos Drakos.

Dear All,

This is to let you know about some changes that will affect the development and support for LaTeX2HTML in the future.

First of all despite the freedom, encouragement and excellent working environment at the Computer Based Learning Unit of the Univ. of Leeds, I have moved on to another position. During the move a large backlog (several hundred) LaTeX2HTML-related messages, queries and bug-reports have accumulated. Apologies to all who have not received any replies...

Future Support

Because of the current workload and the change in circumstances it is no longer possible for me to answer all the LaTeX2HTML queries I receive. I will try to respond as much as possible but in many situations enquirers will receive a standard reply (directing them to the archives and to this mailing list).

LaTeX2HTML Archives

The searchable LaTeX2HTML archives (including messages from this mailing list, comments, bug reports, documentation, etc) will continue to be updated and maintained. They will be kept at the usual address.[15]

Future Development

The development of LaTeX2HTML will continue but at a much slower pace. There are already contributions from others which need to be integrated into the main package. Another obvious candidate is the problem of the postscript to GIF conversion. Also, a project between the Computer Based Learning Unit at Leeds and some other institutions concerning an electronic journal is likely to involve contributions in terms of further enhancements to LaTeX2HTML.

Cheers,

Nikos.

[7] http://www.research.att.com/jsac/
[8] http://www.icbl.hw.ac.uk/projects/isle/Doc.html
[9] http://www.cm.cf.ac.uk/lecture_notes.html
[10] http://lal.cs.byu.edu/cs501/homepage.html
[11] http://www.rrz.uni-koeln.de/REDUCE/
[12] http://www.cwi.nl/cwi/people/Guido.van.Rossum/python-tut/tut.html
[13] http://enemy.gsfc.nasa.gov/cossc/cossc.html
[14] http://cbl.leeds.ac.uk/nikos/tex2html/doc/latex2html/latex2html.html
[15] http://cbl.leeds.ac.uk/nikos/tex2html/doc/latex2html/latex2html.html

LaTeX2HTML ervaringen

Van Handleiding in LaTeX tot Hulp Module op het Internet

Arno Kemperman

Laboratorium voor Analytische Chemie (LAC), Katholieke Universiteit Nijmegen (KUN)
gpstud1@sci.kun.nl

Abstract

Welkom op de digitale snelweg! Steeds meer informatie wordt er aangeboden op het Internet, en met het uitgroeien van het Internet groeit de onoverzichtelijkheid. In dit artikel wordt een toepassing van LaTeX2HTML beschreven waarbij een LaTeX document wordt omgezet naar een reeks goed gestructueerde pagina's op het World-Wide Web. Deze toepassing betreft een document die een handleiding is voor Internet-gebruikers (voornamelijk voor medewerkers van Laboratorium voor Analytische Chemie, KUN, Nijmegen). Er worden wat voordelen en voorbeelden van het LaTeX2HTML gebruik gegeven en hoe je het programma kunt aanpassen naar je eigen wensen. Bekijk de LaTeX documenten ook eens op deze moderne manier en blader eens met de muis door je tekst, getransformeerd tot een digitaal wonder op je scherm. Oftewel: LaTeX2HTML2!

1 Inleiding

De mens heeft altijd behoefte aan communicatie gehad: het uitwisselen van informatie. Een wereld zonder communicatie zou eenvoudigweg niet mogelijk zijn. Steeds weer zoeken we dan ook naar nieuwe en snellere vormen van communicatie om het welzijn van de mens te verhogen. Neem bijvoorbeeld de ontwikkeling van de televisie. De televisie heeft toch wel een cultuurschok teweeg gebracht. Mensen worden graag geïnformeerd over alles wat ze interesseert. De televisie geeft echter niet altijd de gewenste informatie. Dit komt met name doordat de televisie niet interactief is. Je moet het doen met wat je wordt voorgeschoteld.

2 Het Internet en World-Wide Web

Het Internet is zo'n medium waarmee het interactief uitwisselen van informatie mogelijk is. Het Internet is een wereldomspannend netwerk gebruik makend van allerlei standaarden en protocollen om informatie te transporteren tussen enorm veel computersystemen. World-Wide Web (WWW) maakt gebruik van het Internet. Een officiële beschrijving van het WWW project is: 'wereldwijde hypermedia informatie-uitwisseling met het doel een universele toegang te verlenen tot een groot aantal documenten'. De informatie wordt verstuurd via het HyperText Transfer Protocol (HTTP). De documenten zijn geschreven in HyperText Markup Language (HTML).

Het HTTP protocol zorgt voor de snelle communicatie tussen de verschillende computersystemen; m.a.w. tussen vraag (client) en aanbod (de WWW server). Als client gebruik je een zogenaamde Web browser (Mosaic, Netscape) die de informatie toont op het beeldscherm. Het protocol zorgt voor een vrije interpretatie van de HTML-code voor de verschillende browsers, zoals Netscape, voor de MAC, PC en UNIX omgevingen. De exacte opbouw voor de verschillende systemen zal dus verschillen.

Het bijzondere gebruik van (HTML) hypertekst door WWW is met name de mogelijkheid die wordt geboden om een verwijzing te maken naar een document welke zich op een andere computer bevindt. Via de browser is zo'n verwijzing meestal zichtbaar doordat het onderstreept is. Aanklikken met de muis brengt je dan naar het gewenste document ergens op een computer in de wereld. Elk document heeft zijn eigen specifieke URL-naam. Een URL zorgt dus eigenlijk voor een unieke naam voor een WWW-pagina waardoor de browser dit document kan vinden op het Internet. Zo'n URL is opgebouwd uit een aantal componenten:

```
<protocol>://<server>/<locatie>
```

`<protocol>` staat voor het soort protocol voor het verzenden van de informatie (HTTP kan ook bijvoorbeeld FTP aan)
`<server>` staat voor een 'knooppunt' op het net: een server-naam waarmee het computersysteem, waarop het gevraagde document zich bevindt, kan worden gevonden.
`<locatie>` geeft de vindplaats van het document aan op de computer van de server.

Er kan verwezen worden naar andere documenten, ASCII-teksten, gopher- en FTP-adressen, nieuwsgroepen, geluid, filmpjes en plaatjes. Naast deze 'links' biedt WWW nog meer mogelijkheden zoals het opnemen van plaatjes in een pagina, invulformulieren, opsommingen, een nette lay-out met verschillende letterstijlen en sinds kort formules en vergelijkingen, tabellen, figuren met tekst er omheen, achtergrond maken en het veranderen van de kleuren van de tekst. Dit maakt het WWW systeem een interessant en aantrekkelijk medium om informatie te zoeken waardoor het

in korte tijd uitgegroeid is tot de meest gebruikte standaard op de digitale snelweg.

3 LaTeX2HTML

Er zijn een aantal programma's beschikbaar die documenten, gemaakt met je favoriete tekstverwerker, kunnen omzetten naar HTML. Hoewel de HTML-code zeer eenvoudig is (er kan gemakkelijk met behulp van een simpele editor een WWW-pagina worden gemaakt), is het soms toch de moeite waard zo'n programma te gebruiken. LaTeX2HTML is zo'n programma wat LaTeX documenten rechtstreeks kan omzeten naar HTML-documenten die gelezen kunnen worden door de verschillende Web browsers. Het programma is geschreven door Nikos Drakos, Computer Based Learning Unit, University of Leeds.[1]

Laat ik wat redenen geven om LaTeX2HTML te gebruiken:

- Een document is direkt beschikbaar te maken op het Internet via World-Wide Web.
- Hierdoor kan het tijdswinst opleveren.
- Je hoeft de HTML-codes niet te kennen om toch een HTML pagina te maken.
- Het brengt structuren aan in een document zodat bijvoorbeeld stukken tekst snel te vinden zijn ('links' naar hoofdstukken bijvoorbeeld, via een inhoudsopgave of de woorden in de index zijn aanklikbaar).
- Het zet zo veel mogelijk automatisch om naar het HTML formaat: de stijl (bijvoorbeeld grootte), inhoudsopgave, index, formules, voetnoten en speciale tekens.
- Het is mogelijk (hyper)links te maken naar andere adressen/documenten. Zo wordt een document wel erg dynamisch: je referenties zijn bijvoorbeeld direkt bereikbaar door simpel klikken met de muis.
- Je kunt een document nog steeds uitprinten in je ouwe vertrouwde LaTeX lay-out.

4 Een Toepassing, Inleiding en Aanleiding

Begin 1994 ging ik stage lopen voor mijn scheikunde studie op de afdeling Laboratorium voor Analytische Chemie (LAC), Katholieke Universiteit Nijmegen. In die tijd was Mosaic de enige browser die werd gebruikt op de afdeling. Tegenwoordig wordt voornamelijk Netscape gebruikt. In het begin van het gebruik van Mosaic werd het eigenlijk vaker gebruikt voor minder nuttige doeleinden (zoals Interzappen of Web-surfing; Interzappen is voor mij 'het rondzwerven op het Internet op zoek naar nuttige of minder nuttige informatie en soms niet meer weten hoe je er bent gekomen'). Toen was nog niet duidelijk dat het Internet ook zeer nuttige informatie voor voor het wetenschappelijk onderzoek kon opleveren. Denk bijvoorbeeld aan artikelen en informatie over mailing lijsten, Frequently Asked Questions (FAQ's), congressen, nieuwsgroepen en FTP-adressen.

Langzamerhand werden de verschillende toepassingsmogelijkheden van het World-Wide Web ook door de afdeling ontdekt en ontstond de vraag of het LAC zich ook niet moest kunnen presenteren op het Internet. Het LAC kon zo bereikbaar zijn voor onderzoekers in hetzelfde vakgebied en informatie beschikbaar stellen in de vorm van informatie over medewerkers, hun onderzoek en eerder gepubliceerde artikelen. Begin 1995 verden er een aantal WWW-pagina's geschreven die in april'95 officieel op het Internet zijn gepresenteerd.[2]

Mijn bijdrage was voornamelijk het inventariseren en structureren van allerlei verwijzingen naar nuttige adressen op het Internet, met name naar informatie over onderwerpen op de afdeling. Deze onderwerpen zijn voornamelijk chemometrie, kunstmatige intelligentie, neurale netwerken en genetische algoritmen. Ook wordt in een pagina vele verwijzingen gegeven naar informatie over het Internet en WWW zelf, zoals allerlei handleidingen en interessante startpunten. Het opzetten van deze pagina's vereist natuurlijk wel wat kennis over het Internet en hoe en waar bepaalde informatie te vinden is. De opgedane kennis wilde ik in een verslag zetten zodat dit als een handleiding kan gaan dienen voor medewerkers van de afdeling om snel informatie te kunnen vinden over allerlei onderwerpen, en verwijzingen naar WWW-pagina's waar eventueel nog meer informatie over het gewenste onderwerp te vinden is.

De handleiding wilde ik schrijven in LaTeX omdat het een programma was waardoor ik een simpele editor (xedit) in UNIX X-Windows kon gebruiken om mijn teksten te schrijven (handig als je veel 'verft' met je muis, het kopieren van lappen tekst, en met name URL-adressen die toch vaak erg lang en ingewikkeld zijn). Bovendien wilde ik er gewoon wel eens mee leren werken. Met de juiste LaTeX commando's is een fraaie lay-out met een mooi lettertype zo gemaakt. Tijdens een van mijn ritten op de digitale snelweg kwam ik in een document een verwijzing naar het LaTeX2HTML programma tegen.

Dit programma converteert documenten naar het formaat dat geschikt is voor WWW en dit leek me wel geschikt om mijn scriptie om te zetten naar HTML. Wat is mooier dan een handleiding over het Internet op het Internet zelf! Het leek me ideaal om alle verwijzingen naar het Internet aanklikbaar te kunnen maken.
Na het overhalen van het programma via ftp bleek tijdens het installeren dat het programma wel wat andere programma's nodig had om te kunnen werken zoals `perl`, `dvips` en `gs`. Tijdens het zoeken naar deze programma's op het netwerk van de universiteit bleek LaTeX2HTML er zelf al op te staan! Een eerste poging om een LaTeX tekst om te zetten was toen snel gedaan en het resultaat was al verbluffend. Ook een aantal medewerkers op de afdeling zijn nu enthousiast over dit programma. Ik heb wel wat mogelijkheden gebruikt om het programma aan te passen. Deze veranderingen waren simpel uit te voeren door de confi-

[1] Zie ook: `http://cbl.leeds.ac.uk/nikos/tex2html/doc/latex2html/latex2html.html`
[2] `http://www.sci.kun.nl/cac/www-home.html`

guratiefile `.latex2html-init` aan te passen die met het programma wordt meegeleverd. Nikos Drakos heeft bij het programma een prima handleiding geschreven waarmee alle mogelijkheden duidelijk worden uitgelegd. Een aantal toepassingen en aanpassingen aan het programma heeft er uiteindelijk voor gezorgd dat mijn LaTeX documenten worden omgezet naar HTML, aangepast op mijn wensen, terwijl ik het ook gewoon netjes kan uitprinten in LaTeX.

5 Gebruik en Aanpassing LaTeX2HTML Versie 95

Met het programma LaTeX2HTML wordt een voorbeeld configuratiefile `.latex2html-init` (`dot.latex2html-init`) meegeleverd om je eigen instellingen te veranderen. Hieronder volgen de veranderingen en waarom ik die heb gemaakt.

dot.latex2html-init

1. `$MAX_SPLIT_DEPTH = 8;`
   `# Stop making separate files at this depth`
   Veranderd van 8 naar 2 omdat ik overzicht wilde behouden in mijn HTML versie van mijn LaTeX document. Een maximale diepte van 2 zorgt ervoor dat alle `\subsection` delen in 1 HTML pagina kwamen. De hoeveelheid tekst in een subsection vond ik niet groot genoeg om deze in een apart bestand te zetten. Het geheel wordt leesbaarder. In mijn geval wordt een hele `\section` in één keer leesbaar.
2. `$MAX_LINK_DEPTH = 4;`
   `# Stop showing child nodes at this depth`
   Veranderd naar 2. Ook dit om niet onnodig te laten verwijzen naar allerlei sub-paragraafjes, dat maakt het niet overzichtelijker.
3. `$TITLE = "Het Internet";`
   Titel van het document en de startpagina in HTML.
4. `$AUTO_NAVIGATION = 1;`
   Niet veranderd, de navigation gebruik ik om te 'bladeren' in het HTML document naar een volgende en vorige bladzijde etc. Ik heb wel eigen icoontjes gebruikt en de volgorde veranderd.
5. `$CHILDLINE = "<BR> <HR>\n";`
   veranderd naar:
   `$CHILDLINE = "<BR> <IMG SRC= blauwe_lijn.gif><BR>";`
   Geeft dus een lijn (gif-plaatje) in het HTML document in plaats van een gewone lijn (`<HR>`).
6. `$WORDS_IN_PAGE = 1000;`
   Naar 300. Als er meer woorden zijn dan komt onderaan het document ook navigatie. Wel handig omdat je dan niet hoeft terug te scrollen om op het icoontje 'volgende bladzijde' te klikken.
7. `$default_language = 'english';`
   naar:
   `$default_language = 'dutch';`
   Het verslag is in het Nederlands.
8. `$WORDS_IN_NAVIGATION_PANEL_TITLES = 4;`
   Veranderd naar 10. Oorspronkelijk staat de tekstgestuurde navigatie bovenaan de pagina naast elkaar. Hij kapt dan bijvoorbeeld een titel van een hoofdstuk af na 4 woorden, maar aangezien ik het onder elkaar wilde hebben is het afkappen niet echt nodig.
9. `$EXTERNAL_UP_LINK = "http://www.sci.kun.nl/cac/www-home.html";`
   `$EXTERNAL_UP_TITLE = "Terug naar de LAC-page";`
   Deze gewoon ingevuld. Geeft een link op de eerste HTML pagina zodat je terug kunt verwijzen naar het document waarop verwezen werd naar het LaTeX2HTML document (waar je dus vandaan kwam).
10. `$NETSCAPE_HTML = 1;`
    Op 1 gezet, blijkt iets mooier te zijn indien je met Netscape werkt.
11. `$TITLES_LANGUAGE = "dutch";`
    Naar dutch gezet dus.
12. ```
    sub dutch_titles {
     $toc_title = "Inhoudsopgave";
     $lof_title = "Lijst Figuren";
     $lot_title = "Lijst Tabellen";
     $idx_title = "Index";
     $bib_title = "Referenties";
     $info_title = "Over dit dokument...";
    }
    ```
    Dit toegevoegd. In de handleiding bij LaTeX2HTML wordt een voorbeeld gegeven hoe je dit kon aanpassen naar de gewenste taal. Dit voorbeeld overgenomen en veranderd naar het Nederlands.
13. **Navigation Panel**
    ```
    #  Start with a horizontal rule (3-d
    #                              dividing line)
    "<BR> <HR>".
    ```
    Naar:
    ```
    #  Start with a horizontal rule (3-d
    #                              dividing line)
    "<IMG SRC=blauwe_lijn.gif> <BR><BR>".
    ```
    Mooiere blauwe lijn ingevoegd ipv een gewone lijn.
14. ```
    # Now add a few buttons with a space
    #                                between them
    "$NEXT $UP $PREVIOUS $CONTENTS $INDEX
                            $CUSTOM_BUTTONS" .
    ```
    Naar:
    ```
    # Now add a few buttons with a space
    #                                between them
    "$PREVIOUS $UP $NEXT $CONTENTS $INDEX
                            $CUSTOM_BUTTONS" .
    ```
    De volgorde veranderd! Een wat nettere lay-out omdat ik pijlen gebruik in het navigatiepaneel. Door de volgorde `$PREVIOUS $UP $NEXT` te nemen krijg je de pijlenvolgorde: links – boven – rechts, deze verwijzen naar respectievelijk: vorige pagina — 'omhoog in document structuur' — volgende pagina. Er werden geen `$CUSTOM_BUTTONS` gebruikt.
15. ```
    # If ''next'' section exists, add its title
    #                    to the navigation panel
    ($NEXT_TITLE ? "<B> Next:</B> $NEXT_TITLE\n"
                                    : undef) .

    # Similarly with the ''up'' title ...
    ($UP_TITLE ? "<B>Up:</B> $UP_TITLE\n"
                                    : undef) .

    # ... and the ''previous'' title
    ```
    ```

```
($PREVIOUS_TITLE ? "<B> Previous:</B>
                $PREVIOUS_TITLE\n" : undef) .
```

Veranderd naar:

```
# ... and the ‘‘previous’’ title
($PREVIOUS_TITLE ? "<BR> <B> vorige blz:</B>
            $PREVIOUS_TITLE <BR>" : undef) .

# Similarly with the ‘‘up’’ title ...
($UP_TITLE ? "<B> op:</B> $UP_TITLE <BR> "
                                  : undef) .

# If ‘‘next’’ section exists, add its
#               title to the navigation panel
($NEXT_TITLE ? "<B> volgende blz.:</B>
                $NEXT_TITLE <BR>" : undef) .
```

Hier de taal aangepast voor de tekstgestuurde navigatie. Door de `\n` te vervangen door `<BR>` komen de verwijzingen onder elkaar te staan wat ik mooier vond.

16.
```
#  Line Break, horizontal rule (3-d
#           dividing line) and new paragraph
"<BR> <HR> <P>\n"

#  Line Break, horizontal rule (3-d
#  dividing line) and new paragraph
"<BR>  <IMG SRC=blauwe_lijn.gif> <P>\n"
```

Ook hier een mooie scheidingslijn aangebracht.

17.
```
%icons =
(
 'cross_ref_visible_mark' ,'d_hand.gif',
 'anchor_mark' ,' ',
 'anchor_invisible_mark' ,' ',
 'up_visible_mark' ,'cyuarrw.gif',
 'next_visible_mark' ,'cyrarrw.gif',
 'previous_visible_mark' ,'cylarrw.gif',
 'next_page_visible_mark' ,'cyrarrw.gif',
 'previous_page_visible_mark' ,'cylarrw.gif',
 'contents_visible_mark' ,'type_wr.gif',
 'index_visible_mark' ,'note01.gif',
 'footnote_mark' ,'d_hand.gif',
 'up_inactive_visible_mark' ,'cyuarrw.gif',
 'next_inactive_visible_mark' ,'cyrarrw.gif',
 'previous_inactive_visible_mark','cylarrw.gif',
 'next_page_inactive_visible_mark' ,
                                  'cyrarrw.gif',
 'previous_page_inactive_visible_mark',
                                  'cylarrw.gif',
 'change_begin_visible_mark','yl_ball.gif',
 'change_end_visible_mark','yl_ball.gif',
 'change_delete_visible_mark','yl_ball.gif'
 );
```

Aangemaakt in de configuratiefile aangezien ik andere icoontjes wilde gebruiken. Er zijn ook inactieve icoontjes. Deze worden gebruikt als er bijvoorbeeld geen pagina ‘terug’ is. Voor de index en inhoudsopgave zijn ook aparte icoontjes gekozen.

18. Om de icoontjes te laten zien, moet worden veranderd:

```
$ICONSERVER = 'http://ac7.sci.kun.nl:
                          6666/latex/gif';
```

6 Gebruik LaTeX2HTML opties

Algemeen:

```
latex2html [optie] latex.tex
```

Veel van de LaTeX2HTML opties kunnen ook in de configuratiefile `.latex2html-init` worden geregeld zoals:

`-t NAAM`	naam van het document
`-no_navigation 0`	geen navigatie (of 1: wel navigatie)

De enige optie die ik gebruikt heb (dus buiten de configuratie-file) is:

`-dir directory`	directory waar de HTML bestanden moeten worden geïnstalleerd

Voor meer van deze opties verwijs ik weer naar de handleiding van Nikos Drakos.

7 Gebruik Extra Commando's in LaTeX

Voor LaTeX is een extra `.sty` bestand beschikbaar: `html.sty`. Deze bevat informatie over extra toe te voegen commando's met betrekking tot het toevoegen van extra HTML mogelijkheden. Een aantal daarvan heb ik gebruikt:

1. Toevoegen `html.sty`:

```
\documentstyle[dutch,epsf,html]{article}
```

2. De hyperlink ‘Tekst’ wordt onderstreept en aanklikbaar in HTML en als een voetnoot wordt `http://lycos.cs.cmu.edu/` afgedrukt in LaTeX:

```
\htmladdnormallinkfoot{Tekst}{http://}
                                lycos.cs.cmu.edu/
```

3. Hetzelfde als hierboven, alleen wordt nu geen voetnoot gegeven:

```
\htmladdnormallink{Tekst}{http:}
```

4. Plaatje invoegen in HTML door verwijzing naar een locatie (URL), wel onhandig als je het plaatje ook wil gebruiken voor LaTeX. Dan moet je echt de normale LaTeX commando's gebruiken en bijvoorbeeld het plaatje als PostScript in je directory zetten (LaTeX2HTML zet deze netjes in je WWW pagina met onderschrift etc.):

```
\htmladdimg{URL}
```

5. Begin en einde ‘platte’ html-code, handig als je hele stukken HTML-code hebt. Deze worden gewoon in de WWW pagina tussengevoegd. Heb ik gebruikt om hier en daar wat plaatjes toe te voegen voor een leuke lay-out. LaTeX zelf doet niets met deze code:

```
\begin{rawhtml}
\end{rawhtml}
```

6. Begin ‘platte’ LaTeX-code. Deze wordt niet omgezet naar HTML, wel naar LaTeX uiteraart:

```
\begin{latexonly}
\end{latexonly}
```

7. Hyperlink naar een referentie:

```
\htmlref{tekst}{fig:example}
```

8. Eigen button naar een eigen locatie, dus invullen:

```
\htmladdtonavigation
      {\htmladdnormallink
              {\htmladdimg{http:?.gif}
      }
      {http:}
```

Figuur 1: *Voorbeeld van een LaTeX2HTML pagina layout. Bovenaan het navigatiepaneel. De pijlen verwijzen naar een volgende of vorige bladzijde of naar een niveau hoger in het document. Daaronder de aanklikbare titels.*

Figuur 2: *Voorbeeld van een stuk tekst in HTML waarbij de onderstreepte woorden links zijn naar referentiedocumenten. Deze links zijn in LaTeX aangemaakt door de speciale commando's in html.sty.*

8 Slot

LaTeX2HTML gebruik ik nu dagelijks.[3][4] Zo kan ik zien hoe mijn document er in hypertekst-vorm uit ziet en probeer ik het zo gebruikersvriendelijk mogelijk te laten zijn. Doordat een aantal hoofdstukken een eigen URL hebben is het nu mogelijk te verwijzen naar dit hoofdstuk als een hulp-optie bij de verschillende onderwerpen op de LAC pagina op het Internet.

Ik wil ook nog wijzen op de mogelijkheden die LaTeX2HTML biedt voor niet-Internetters. Met de verschillende browsers kunnen ook bestanden bekeken worden via de zogenaamde localhost (via een `file://localhost/..` of `file:/..` URL). Dit geeft de mogelijkheid om je document zelf door te lezen. Als een soort luxe alternatief om je LaTeX document door te nemen. Met name de mogelijkheid dat je hoofdstukken direkt aan kunt klikken, werkt naar mijn mening zeer prettig en geeft je een prima overzicht over je document (vooral als dat zeer groot is). Ik gebruik het zelf nu om mijn teksten makkelijk na te kunnen lezen zonder ik met mijn ogen een weg moet kunnen banen tussen al die LaTeX commando's (het is daarnaast ook mooier dan bijvoorbeeld Ghostview). Al met al heb ik hierboven veel veranderingen aangeven, maar het beste is gewoon om het zelf eens uit te proberen: trial and error!

[3] LaTeX2HTML pagina: `http://www.sci.kun.nl/cac/latex/scriptie/scriptie.html`
[4] De LAC Home Page: `http://www.sci.kun.nl/cac/www-home.html`

Electronic Publication and Data Distribution for the Five College Astronomy Department

Karen M. Strom

Five College Astronomy Department,
University of Massachusetts,
Amherst, MA 01003, USA
kstrom@hanksville.phast.umass.edu

Abstract

The Star Formation Group at FCAD has begun to make use of the World Wide Web to explore the advantages of hypermedia presentations for the distribution of preprints, Ph. D. theses and observatory publications. As a byproduct of this work, a method for displaying subscripts (*e.g.* $M_{\odot}$, $\Delta\nu_D$) and superscripts (*e.g.* T^4, $C^{18}O$) has been developed.
This set of bitmaps is publically available.

1 Introduction

Most of the professional journals in astronomy are now accepting submissions in LaTeX. It is clear that advances in manuscript preparation have not been matched by increases in distribution speed — the first and most easily achievable gain which should result from some standardization of the format for manuscript submission.

There are enormous advantages to having papers available electronically with *their associated data sets.* They can be made available almost instantly upon the completion of the refereeing process. They are easily searchable, and, if the tabular material is in an HTML document (if small) or other easily interpretable form, the data is instantly accessible. Large data sets can be made available in standard, generally accepted formats. More sophisticated graphics are easily and cheaply included. Links can be built into the paper to references and abstracts that are on-line as well as to other non-standard materials not usually available to readers of the papers.

2 Preprints and Ph.D. Theses

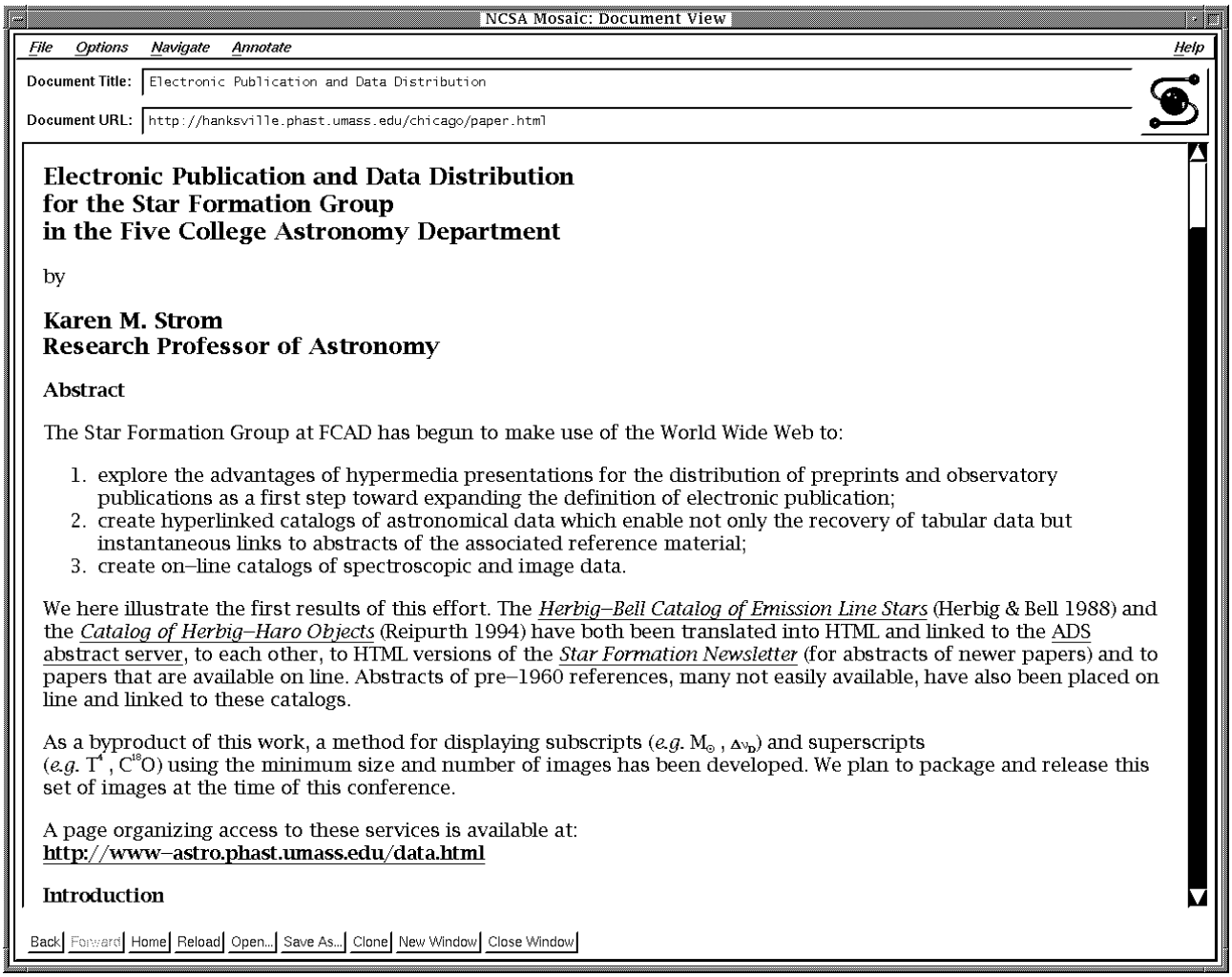
Since astronomy journals are now accepting manuscripts in LaTeX, the obvious method for conversion of these manuscripts into HTML is by use of Nikos Drakos' [1] LaTeX2HTML. This *perl* script will translate a LaTeX manuscript into a set of linked HTML pages, translating the mathematical expressions into transparent GIFs, placing these images in the text as in-line images. However LaTeX2HTML may create many copies of the same image when the same character is reused in the document, particularly for commonly used symbols and expressions of units, such as cm^2 or s^{-1}. Since all images were installed with `ALIGN=BOTTOM`, *e.g.* cm2 or s-1, the resulting text could be difficult to read. It is also true that math mode expressions including subscripts would appear to float above the rest of the text on the line,balanced on the subscript *e.g.* $T_{\rm eff}$. Of course, because these expressions were inserted in the text as images, the font would rarely match that of the rest of the text.

In response to these practical and perceptual problems, I developed an image library of the most commonly used subscripts, superscripts and mathematical symbols used in the manuscripts which I had converted to HTML. This library was designed with several considerations in mind.

- The symbols should blend as inconspicuously as possible with the text.
- The total size of the images to be transferred should be as small as possible to minimize load time and network traffic.
- The reading of an entire paper should be accomplished without need to discard and reload either symbol or figure images if at all possible.

The first item on this list concerns the fact that, when the entire expression is captured as an image, with default font settings, as in LaTeX2HTML, the expression tends to look out of place in the rest of the text generated by the browser, no matter which font is selected. However, if only the sub- and superscripts are generated as bitmaps, this problem is greatly alleviated. Many times letters are the main symbols used and are therefore simply part of the client generated text. The sub- and superscripts are always generated in another font and are much smaller so that few differences in the structure of the character are possible or noticeable.

The last two items take into consideration the default image cache size of most browsers, and the fact that the least recently accessed image will be the first discarded. The typical size of a 2 – 5 color image used as a figure in these preprints is 4 – 8 kilobytes. The typical size of a bitmap to be used as a sub- or superscript is 0.06 kilobytes. Thus our third consideration should easily be met.



These three considerations thus work in concert to provide a clean-looking, easily readable paper, taking minimal time to load each page, thus allowing browsing as well as reading in depth without forcing the scientific community to adopt another, temporary, shorthand for complex mathematical expressions or simply delaying the onset of electronic publication of journal papers.

The library includes the Greek letter set previously made available, the numerals, the entire alphabet, both capital and lower case, and some special symbols and letter combinations of common use in astronomy. To implement this set of transparent GIFs, *perl* scripts were written to preprocess the manuscripts to insert the in-line images as required. LaTeX2HTML is then used to convert the manuscript to HTML pages and a post-processing *perl* script is used to clean up the few things that may have been affected by LaTeX2HTML. The manuscript is then ready for insertion of the images, tables and references.

As a sidelight, I note that the display of subscripts is a simple use of the HTML tag for displaying an in-line image. The unexpected *subscript* effect occurs because, in this case, the image is smaller than the text height. Thus when you specify

```
<IMG ALIGN=MIDDLE ALT="_sun"
                  SRC="/kicons/smsun.gif">,
```

the middle of the small image is aligned with the base line of the text. Thus a subscript, $M_{\odot}$, is displayed. This method is also used to place the Greek letters having trailers properly on the baseline. The images are filled out with sufficient blank pixels below the letter that setting `ALIGN=MIDDLE` will properly locate the letter. I also urge users of this library to always make use of the `ALT = "xxxx"` option so that people accessing your files with **Lynx** may more easily read your pages. This is easily accomplished when the substitutions are made in the manuscript by the *perl* script.

Although newer versions of LaTeX2HTML allow the option of conversion of Postscript figures, at a user-specified scale, into transparent GIFs in-lined into the text, I prefer to exercis more control over the scaling and appearance of the figures because the figures were initially designed for display on paper.[1] I display the file using *ghostview* and then scale the image to the desired size, then capturing the image using *xv*. For some images, in particular, gray-scale

[1] See my online tutorial on using images on the World Wide Web at:
`http://hanksville.phast.umass.edu/~kstrom/tutorials/images.html`

and full color Postscript files, ImageMagick may be better for this conversion process. I have chosen to place the scales, axis labels and other labeling text in a deep blue to separate it from the text. However, although the figure on the printed page may be very small, and we are not constrained by space considerations on the HTML page, the spatial resolution available to us is still less than that in the paper representation. Therefore I have chosen to make use of the fact that color is an option available at no extra cost to the server. When different line styles or point styles are called for, different colors are substituted by making use of *xpaint*. When the desired changes are completed, the file is converted into a transparent GIF and placed in the HTML document at the appropriate location. There is much more flexibility in figure placement in HTML documents as one is not forced to manuever within fixed page sizes. For this reason, figures may be more reasonably located with respect to their textual references [2].

There are certainly some very complex figures for which this technique will not be appropriate. In that case a thumbnail or postage stamp image can be made, either by downscaling the entire image or by cropping a recognizable section from the image for insertion into the text. This smaller image can then act as a link to the Postscript (or GIF) high resolution figure.

Tabular material presents different problems. Small tables can be placed within an HTML document by using the `<PRE>` tag. Larger tables are only practical, at the moment, as Postscript files linked to the text at the appropriate point (but see the Catalogs section below). The use of HTML 3 should allow easy introduction of tabular material into the text. Very large tables can be associated with papers in other formats more appropriate to the data.

The Ph.D. theses can be found at:
`http://decoy.phast.umass.edu/`
The preprints from the star formation group may be found at:
`http://www-astro.phast.umass.edu/sfpreprints.html`

The library of transparent GIFs may be found at:
`ftp://hanksville.phast.umass.edu/pub/misc/newgifs.tar.Z`

A more complete paper describing their use is found at:
`http://hanksville.phast.umass.edu/chicago/paper.html`

References

[1] Drakos, N. The manual for LaTeX2HTML is available from:
`http://cbl.leeds.ac.uk/nikos/tex2html/manual/manual.html`

[2] Tufte, Edward R. 1983, *The Visual Display of Quantitative Information*, Graphics Press, Cheshire, CT.

A World Wide Web Interface to CTAN*

Norman Walsh

O'Reilly and Associates
90 Sherman Street
Cambridge, MA 02140
Phone: (617) 354-5800; Fax: (617) 661-1116
norm@ora.com

Abstract

There are a lot of different software packages, style files, fonts, etc. in the CTAN archives. Finding the things you need in a timely fashion can be difficult, as I found out while writing *Making TEX Work*. The ability to combine descriptions of packages with the directory listings from CTAN could help alleviate some of the difficulty. HTML, the document structuring language of the World Wide Web, provides one possible means of combining different views of the archive into a single vision. The CTAN-Web project is my attempt to provide this vision.

1 Introduction

A functioning TEX system is really a large collection of programs that interact in subtle ways. Processing even a relatively simple document like this one requires several programs (TEX, a previewer, and a printer driver at the very least), most of which read input files or can be configured in other ways. It was this complexity that led me to start writing *Making TEX Work* [5], a book I hoped would unravel many of these intricacies.

In the process of writing *Making TEX Work*, I looked at a lot of the software packages, style files, fonts, etc., in the CTAN archives. It really made me appreciate how much stuff the TEX user community has made freely available. By my estimates there are more than 31,000 files in more than 2,300 directories in `/tex-archive` on `ftp.shsu.edu`.

My first challenge was to find the things that I wanted to write about. This was a long process that involved coordinating (at least mentally) the lists of files in the upper-level CTAN directories, entries from David Jones' `TeX-index`, descriptions maintained by the CTAN archivists, my own intuitions about what was available, and the tidbits that I had collected over the years from `Info-TeX` postings. It was occasionally tedious, but it was never really difficult (at least technically).

When the book was beginning to fall into place and I was starting to try to track down all the loose ends, I came to a realization: in the early days, finding things had been an end as well as a means. Now, with pressure mounting on an almost daily basis to finish, I discovered just how hard it was to find things on CTAN. This is not a criticism of the CTAN archivists in *any* way. Without their foresight and diligent efforts, the task could easily become impossible. It's just a fact: there's a *lot* of stuff out there.

One tool became invaluable in my daily efforts: ange-ftp for GNU emacs. GNU emacs, if you aren't familiar with it, is an extremely flexible and powerful editor (it's most common on UNIX workstations, but versions exist for MS-DOS, Windows, OS/2, VMS, and a few other platforms). One of the editing modes of emacs, called *dired*, allows you to 'edit' directories (a directory listing appears in a window on the screen). In dired mode, the editing keys let you rename, copy, delete, view, and edit files, among other things. Ange-ftp is an extension for emacs that lets you edit *remote* file systems via FTP in dired-mode. This lets me load the `/tex-archive/macros` directory from `ftp.shsu.edu` into an emacs buffer and view files simply by pointing to them and pressing 'v.' Ange-ftp handles all of the transactions with the FTP client in the background. Ange-ftp made gathering information from README files *much* easier.

2 Inspiration

What I really wanted wasn't an easier way to browse directories, no matter how grateful I was to have that, but a way of combining the `TeX-index` and other descriptions with a directory listing in some coherent way. A typical interaction with CTAN, in my experience, goes something like this: I need a *widget*, that's under the *something* directory. Oh! There are several things like that. This one looks interesting. Nope that's not it. How about this one. Yeah, that's better. Still, is this other one better? Nope. I'll try the second one.

I find this sort of interaction tedious via FTP.

*Presented at TUG'94, the 15th Annual TEX Users Group Meeting, July31st – August 4th, 1994, Santa Barbara, California, USA. Published in TUGboat **15.3**, 339–343 (1994). Reprinted with permission.

As it happens, I was also beginning to explore the World Wide Web (WWW) at the same time, motivated, in part, by experimentation with LaTeX2HTML and other tools that translate TeX documents into HTML for online documentation projects. Might this be the answer, I wondered. . .? After several days of hacking, the first incarnation of CTAN-Web was born; the CTAN-Web home page is shown in Figure 1.

3 What is the World Wide Web?

The WWW is a vast collection of network-accessible information. In an effort to make this information manageable, protocols have been developed for cross-referencing the Web and software written to browse documents in the Web. One of the most popular browsers is Mosaic, a browser from the National Center for Supercomputer Applications (NCSA).[1] WWW documents use hypertext to make traversing between documents transparent, allowing the user to follow a stream of ideas without regard to where the embodiment of the ideas exists in the Web.

Hypertext links allow you to build dynamic relationships between documents. For example, selecting a marked word or phrase in the current document can display more information about the topic, or a list of related topics.

Naturally, WWW documents can contain hypertext links to other WWW documents, but they can also contain links to documents available through other servers — for example, *Gopher* servers and anonymous FTP servers. Documents in the WWW are addressed by a 'universal resource locator' (URL) that identifies the site from which they are available and the protocol that should be used to retrieve them. The general format of a URL is *protocol://site/pathname*. For example, the URL for the LaTeX help file that I maintain is `http://jasper.ora.com/texhelp/LaTeX.html`; in other words, it is available via the *http* protocol at `jasper.ora.com` in the file `/texhelp/LaTeX.html`.

Once documents are retrieved, it is up to the browser to determine how they should be displayed. In addition to displaying HTML documents directly, many browsers can automatically spawn external viewers to view PostScript documents and image files in a variety of formats.

4 What is HTML?

WWW documents are plain ASCII files coded in HTML [1]. HTML stands for HyperText Markup Language; it is a way to describe documents in terms of their structure (headings, paragraphs, lists, etc.). HTML is really a particular instance of an SGML document. SGML is the Standard Generalized Markup Language and it is defined by the ISO 8897 specification.

The relationship between SGML and HTML can be a little confusing. SGML provides a general mechanism for creating structured documents. HTML documents are SGML documents that conform to a single, fixed structure. (The HTML specification is available at `http://info.cern.ch/hypertext/WWW/MarkUp/MarkUp.html`.)

A detailed exploration of structured documentation principles is beyond the scope of this article, however, a few words may help clarify the picture; users familiar with LaTeX are already familiar with structured documentation.

The key notion is that structures (characters, words, phrases, sentences, paragraphs, lists, chapters, etc.) in a document should be identified by *meaning* rather than appearance. For example, here is a sentence that you might find in an installation guide (this sentence is coded in TeX):

```
Use the {\bf cd} command to change to the
{\it /usr/tmp/install} directory.
```

The same sentence might be coded in a structured way like this:

```
Use the <command>cd</command> command to
change to the <directory>/usr/tmp/install
</directory> directory.
```

The advantage of the structured document is that it is possible to answer questions about the *content* of the document. For example, you might check to see if all of the commands that are mentioned in the installation guide are explained in an appendix. Since commands are explicitly identified, it is easy to make a list of all of them. In the unstructured case, it would be very difficult to identify all the commands accurately.

You can achieve structured documentation in TeX with macros, but you are never forbidden from using lower-level commands. The advantage of using a formal structured documentation system, like SGML, is that the document can be validated. You can be sure that the document obeys precisely the structure that you intended. The disadvantage of a formal system is that it must be translated into another form (or processed by a specialized application) before it can be printed, but that is becoming easier. In the case of HTML, many browsers already exist.

Since an HTML document is described in terms of its structure and not its appearance, most HTML documents can be effectively displayed by browsers in non-graphical environments. There is a browser for Emacs called W3 and a browser called Lynx for plain text presentation, for example.

5 What is CTAN-Web?

CTAN-Web is a collection of WWW documents that combines descriptions of many packages available from CTAN with pointers to each of the files in the archive. At present, the descriptions come from an early draft of my book, David Jones' `TeX-index`, and the `00Description` files in the archives. Over time, additional descriptions will be added. Figure 2 shows the top of the `/tex-archive/macros` directory. In this figure you can see how files and directories in the archive are displayed with their descriptions.

[1] The figures in this paper are of the X11 version of NCSA Mosiac.

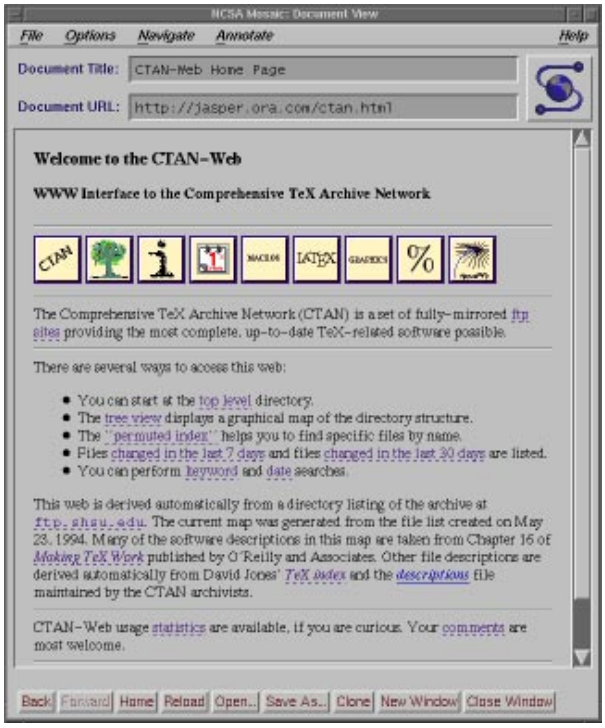


Figure 1: *The CTAN-Web home page.*

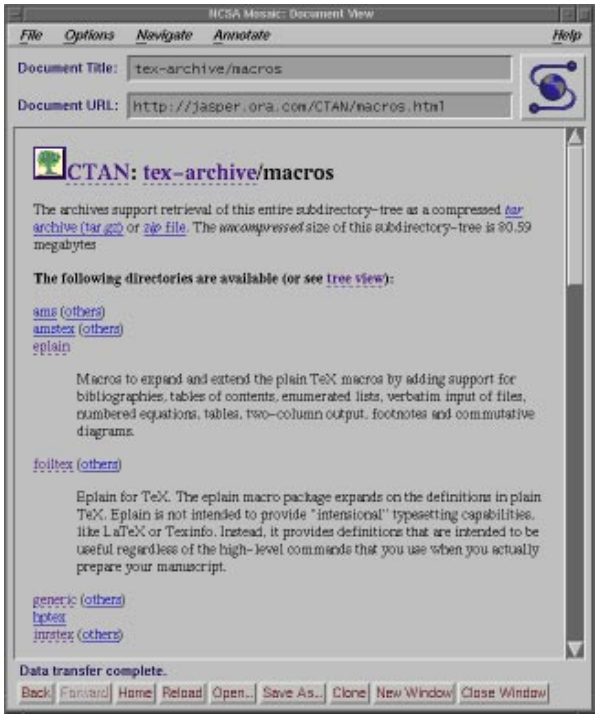


Figure 2: *The* `CTAN:/macros` *directory.*

The CTAN-Web also has the following features:

- Links are made directly to other online references in the Web. For example, the online help files provided in the `info/htmlhelp` directory are also available as WWW documents on the net. This fact is exploited in the descriptions of these files by creating a hypertext link directly to the online help.
  In addition, font samples can be displayed for several METAFONT fonts.[2] Each font sample is stored as a single graphic image and can be viewed with any browser that can display GIF images.
- The CTAN-Web documents are indexed. Users can perform online queries for material based upon any word that appears as a filename or in the online description of any file. Simple conditional searches can also be performed ('x or y' or 'x and y').
  Suppose, for example, that you are looking for a macro file that implements a verbatim text environment for Plain TEX. A combined query for 'verbatim and plain' reveals that the descriptions of 5 files and 9 directories contain both the words 'verbatim' and 'plain'.[3] A combined search can greatly reduce the number of files you must examine. In this case, a query for either 'verbatim' or 'plain' by itself results in more than 20 hits.
- Each instance of a file that appears in more than one place in the archive is identified. For example, any reference to the file `verbatim.sty` identifies all 7 instances of it in the archive.
- Want to know which files were modified within the last 12 days? Or between 1 Jan and 31 Jan of 1993? Information about the age of each file is maintained in a separate database, accessible from the CTAN-Web home page. This allows you to perform online queries of the archive by age.
- A 'permuted index' is constructed each time the Web is built. This allows you to quickly locate files by name.
- A list of files added or modified in the last 7 or 30 days is also constructed.
- A tree (hierarchical) view of the archive is also available. The tree view provides a fast means of 'walking' down into the lower levels of the archive.

6 Reaching CTAN-Web

You can reach the CTAN-Web pages by pointing your browser at the URL:

`http://jasper.ora.com/ctan.html`

7 Behind the Scenes

The Web is now rebuilt on a daily basis using the most recent information from the `ftp.shsu.edu` server. My task in creating the CTAN-Web was organizing the descriptions that I had available and writing the scripts (in Perl, mostly) that construct the web automatically. For those who are curious, this section provides a brief description of the various parts of the Web and how they are linked together.

[2] Samples for all the METAFONT fonts will be generated shortly.
[3] In the Web built on 20 May 1994.

```
<!-- tex-archive/info/htmlhelp/latex-help-html.zip -->
An HTML version of the LaTeX help file created by George Greenwade.
This is the version provided online at <tt>jasper.ora.com</tt>.
It is also available in VMS format (formatted ASCII),
TeXinfo format, HTML format, and as a Microsoft Windows help file.
<!--ONLINE-->
<P>
The LaTeX help file is also
<A HREF="http://jasper.ora.com/texhelp/LaTeX.html">available online</a>.
<!--/ONLINE-->
```

Figure 3: *The description of* `latex-help-html.zip` *in the Web sources.*

7.1 Handling the descriptions

In order to quickly locate descriptions for the various packages, I maintain the collection of descriptions in a directory structure that parallels the CTAN archives. Each description file is written in a mixture of TEX and HTML (a mixture is used so that it may one day be possible to produce a printed version of the Web). For example, the current description of `latex-help-html.zip` is shown in Figure 3.

7.2 Retrieving files from the archives

One of the first problems that had to be solved was how files would be retrieved from the archives. While it's easy to create a link to a file at an FTP site, in the case of CTAN-Web that isn't sufficient because CTAN exists at several sites. The link really needs to be made to the *closest* FTP site.

Although I suppose it is possible to identify the closest FTP site from the user's host id, that seemed impractical. The following compromise was selected instead: rather than linking files directly to an FTP site, they are linked to a script. The document server (`httpd`) provides a facility for making links that cause a program to be executed; the output produced by this program is then displayed as a WWW document. By passing the name of the file requested by the user as an argument to the script, it was possible to write a retrieval script that dynamically constructs a 'retrieval document.' The retrieval document contains links to the requested file at each of the CTAN hosts. It is then possible for the user to select the closest host. An example of the retrieval document created for `README.archive-features` is shown in Figure 4. The primary hosts are shown first, followed by other mirrors that may be closer but are not known to be as reliable.

Selecting a link within the retrieval document causes the browser to actually retrieve the file via anonymous FTP from the selected site.

7.3 Documents in the Web

There are three kinds of documents in the CTAN-Web and within each document there are several kinds of links.

Directory Documents

There is one directory document in the Web for each directory in the archive. Each directory document lists all of the files in the directory it represents along with their associated descriptions.

Directory names in each document are linked to the corresponding directory documents. File names are linked to filename documents (described below) or to the retrieval script, depending on whether the file occurs multiple times in the archive.

The directory document for the `tex-archive/macros` directory is shown in Figure 2.

Tree Documents

There is one tree document in the Web for each directory in the archive that contains subdirectories. The tree document displays three levels of hierarchy starting at the directory it represents.

Directory names in each document are linked to the corresponding tree document. If a directory in the tree does not have subdirectories, it is linked to its directory document instead.

The tree document for the `tex-archive/macros` directory is shown in Figure 5. The tree view provides a larger picture of what exists in a particular directory.

Filename Documents

There is one filename document for each file that occurs in more than one place in the hierarchy. The filename document lists all of the instances of the filename.

Each instance of the filename in the document is a link to the directory document where that file resides in the archive.

The filename document for the `verbatim.sty` file is shown in Figure 6. By identifying multiple copies, the user is given an opportunity to select the file that is most likely to suit his or her needs (the one developed for the format that is being used, for example).

7.4 Building the Web

Early versions of the Web were constructed from the `FILES.byname` list from `ftp.shsu.edu`. Several *Perl* scripts manipulated the listing and constructed the Web.

After a few weeks, it became clear that the `FILES.byname` listing was insufficient for constructing the Web because the list contains no indication of symbolic links, for example. It is also poorly organized for my purposes (the necessity of making multiple passes was causing memory problems). George Greenwade kindly

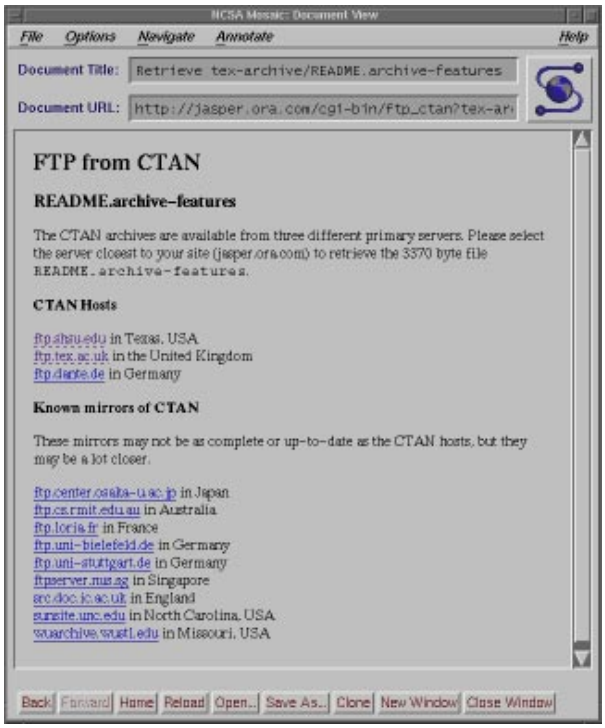


Figure 4: *The retrieval document created for* `tex-archive/README.archive-features`.

Figure 5: *The tree document for* `tex-archive/macros`.

Figure 6: *The filename document for* `verbatim.sty`.

agreed to run a script on the archive that extracts more information and stores it in a form that can be translated into CTAN-Web in a single pass. (This information is provided in `/tex-archive/help/ctan/ctan-web.data.gz`, if you're interested).

8 Room for Improvement

I plan to improve CTAN-Web in a number of ways.

- One of the most important improvements that I have planned is getting the Web off of `jasper.ora.com` and distributed among the CTAN hosts. The Internet connection from `jasper` to the outside world is actually quite slow, and many users find that the performance is poor.
- Assign fixed URLs to each CTAN directory. At present, most of the URLs are assigned more-or-less sequentially when the web is constructed. This means that the URL for the `tex-archive/macros/latex2e/contrib` directory, for example, changes over time. This prevents people from saving the URLs of frequently visited regions of CTAN-Web. The top-level directories already have fixed names.
- Clean up the descriptions. Using an automatic tool to extract the descriptions from several sources in the archives was a fast way to get a large number of descriptions, but the process was not error free. A small, but significant, number of files in the web have incorrect descriptions.
- Potentially add a report-generating function that can return an annotated list of the files that match a particular query.

9 Conclusion

Thousands of documents have been retrieved from the CTAN-Web since it became operational in March, 1994. On the whole, the comments that I have received reinforce my belief that it is useful and flexible way to search the archives.

References

[1] Peter Flynn. How to write html files. `http://www.ucc.ie/info/net/htmldoc.html`
[2] Charles F. Goldfarb. *The SGML Handbook*. Oxford University Press, 1990.
[3] Eric van Herwijnen. *Practical SGML*. Kluwer Academic Publishers, 1990.
[4] Norman Walsh. Ctan-web home page. `http://jasper.ora.com/ctan.html`.
[5] Norman Walsh. *Making TEX Work*. O'Reilly & Associates, 1994.

An Introduction to HyperTEX

Arthur P. Smith

Dept. of Chemistry, BG-10, University of Washington,
Seattle, WA 98195, USA
asmith@mammoth.chem.washington.edu

Abstract

The popularity of the World-Wide-Web and the HTML hypertext language for electronic distribution of information over the internet has thus far been difficult to extend to scientific documents, in part because TEX, the word processing language of choice for technical and mathematical documents, does not have the hypertext capabilities necessary for full integration into the WWW. HyperTEX consists of a collection of extensions of TEX to handle both internal and external (local or networked) hypertext, thus allowing the huge body of TEX-based documents to be more fully integrated with the WWW. This introduction was originally prepared for the American Physical Society e-print meeting at Los Alamos, New Mexico, Oct. 15, 1994.

1 Introduction

The past year has seen a revolution in the processes of Internet-based information navigation and retrieval with the advent of easy-to-use graphical browsers (in particular Mosaic) based on the World-Wide-Web (WWW). The revolution is a result of two components — first the browsers allow a near-uniform (point-and-click or other method) access to documents in almost any format (interpreted according to local `.mailcap` files) and from almost any Internet-based source, accessed as regular files or via ftp, gopher, http or one of many other possible methods, and along with this the Universal Resource Locator (URL) mechanism provides a surprisingly easy and uniform way to specify the location of any document on the net. Second, for certain classes of documents (HTML files, or gopher text files) embedded URL's or other addresses are understood to refer to other, external, documents which can be followed according to the interests of the person viewing the document, producing an interconnected web of documents.

The goal of the HyperTEX collaboration is to extend this second privileged class of documents to include documents based on TEX , the word-processing language of choice for mathematical and scientific writing, thus fully incorporating TEX documents into the burgeoning *web* of information on the Internet.

2 Why HyperTEX?

There already exists one approach for incorporating TEX documents more fully into the *web* — conversion to HTML, as in the program LATEX2HTML by Nikos Drakos. This can work very well, and is already used in some of the electronic publications in mathematics, but there are also several serious problems with this, aside from the technical issues associated with the complexity of the conversion process. HTML by design allows very little author control of the visual form of a document. This is touted as an advantage because it preserves only the *essential* elements of a document and not the artificialities of a page — in fact HTML documents do not have pages at all, although some of the sense of a *page* is implied by separation of a single document into many files. Aside from loss of author control, there is a practical problem of a lack of mathematical tools in the current implementations of HTML — tables and equations are either difficult to implement or impossible. LATEX2HTML gets around this by conversion of such things to bitmapped images, but this is an inefficient and expensive process — and goes in just the opposite direction of HTML's theme of extracting the *essence* of a document, making the document essentially unreadable without a good network connection and a computer with a high quality display.

These problems with HTML are compounded if scientific authors attempt to write documents directly in HTML rather than using TEX first — the lack of authoring tools, the absence of macro capabilities, and the ill-defined nature of the language make this an unpleasant task — just dealing with ordinary text is easy, but getting Greek letters, mathematical symbols, equations and tables into your document is not. The one nice feature of HTML is the ease with which figures can be incorporated into a document. But at least PostScript figures can be incorporated into a TEX document with equal ease using modern DVI interpreters, and the HyperTEX standard presented here allows arbitrary images and other external documents to be referred to and brought to the screen with a single mouse click.

The point of all this is that hypertext capabilities, and the use of URL's to locate new documents — the main feature of HTML that makes it such a useful network information navigation tool — can be much more easily incorporated into TEX than the mathematical capabilities of TEX and the years of experience embedded in various TEX macro

packages can be incorporated into HTML. Whether TEX in general provides a better model for the viewing of on-line information remains to be seen.

3 How does it work?

The underlying element of our implementation of HyperTEX is the use of a TEX macro that bypasses the TEX interpretation process and sends a message directly to the DVI interpreter that processes TEX output. This is the `\special` macro, previously used to define procedures for drawing or including figures in TEX documents. When the characters `\special{string}` appear in the TEX document, the *string* is passed directly without interpretation to the output DVI file (preceded by a marker to identify this as a *special* message to the DVI interpreter). The DVI previewers or processers then interpret this string according to its first few characters. The original HyperTEX specification (due to Paul Ginsparg, Tanmoy Bhattacharya, and me) uses the initial characters *html:* to denote HyperTEX elements in an HTML-like style. David Oliver (`oliver@gang.umass.edu`) has introduced a slightly different specification that uses the initial characters *hyp* to denote his own style of HyperTEX. I will discuss only the original specification in this paper, since as far as they are currently implemented both specifications are essentially equivalent. Note that DVI interpreters that do not understand the *html:* or *hyp* special commands will ignore them, or at worst print out warning messages. Therefore DVI files processed to include HyperTEX commands are fully compatible with old DVI interpreters.

After the initial *html:* string, the specification is identical to a restricted form of HTML. The five arguments we have added to the `\special` command are:

href: `html:<a href = "href\_string">`
name: `html:<a name = "name\_string">`
end: `html:</a>`
image: `html:<img src = "href\_string">`
base_name: `html:<base href = "href\_string">`

The *href*, *name* and *end* commands are used to do the basic hypertext operations of establishing links between sections of documents. The *image* command is intended (as with current html viewers) to eventually place an image of arbitrary graphical format on the page in the current location. Currently for XHDVI, *image* brings up an external viewer with the image, if such a viewer is available. The *base_name* command should be used to communicate to the DVI viewer the full (URL) location of the current document so that files specified by relative URL's may be retrieved correctly.

The href and name commands must be paired with an end command later in the TEX file — the TEX commands between the two ends of a pair form an *anchor* in the document. In the case of an href command, the *anchor* is to be highlighted in the DVI viewer, and when clicked on will cause the scene to shift to the destination specified by *href_string*. The *anchor* associated with a name command represents a possible location to which other hypertext links may refer, either as local references (of the form `href="#name\_string"` with the *name_string* identical to the one in the name command) or as part of a URL (of the form *URL#name_string*). Here *href_string* is a valid URL or local identifier, while name_string could be any string at all: the only caveat is that '"' characters should be escaped with a backslash ('\'), and if it looks like a URL name it may cause problems. There may also be problems if LATEX tries to interpret the *href_string* or *name_string* — in that case preceding the command with `\protect` should usually work. Any defined *name_string* can be referred to in any href referring to the document, in the form `href="URL#name_string"`. Note that anchors may be nested. The only restriction in current implementations is that anchors are truncated at page boundaries.

Because this html-based naming scheme is somewhat unwieldy, although very general, Tanmoy Bhattacharya (`tanmoy@qcd.lanl.gov`) has written several collections of TEX macros to simplify things. The basic package is *hyperbasics.tex* which defines the following simple low level hypertex macros:

- `\href{url}{text}`: text becomes an href anchor referring to *url*.
- `\hname{myname}{text}`: text becomes a name anchor with name *myname*.

plus others that are used to automatically convert LATEX or other style markup into corresponding names and references.

4 How do I use it?

4.1 As a reader

There are currently three DVI interpreters that understand the HyperTEX `\specials`: XHDVI for X windows, HYPERTEXVIEW.APP for NextStep, and DVIHPS for conversion to a 'hyper' postscript that includes PDFmark operators for further conversion to PDF, or for interpretation by appropriately modified postscript previewers. We hope to encourage others to add HyperTEX capabilities to DVI previewers or TEX authoring tools for Macintosh or IBM-PC systems.

For a TEX document that has already been processed to a DVI file with HyperTEX elements, viewing the internal hypertext is almost trivial — you just start one of the interactive DVI previewers and navigate by button clicks as with Mosaic or other WWW browsers. To have XHDVI, for example, brought up automatically from Mosaic when a DVI document is referenced, you need to have a *.mailcap* file in your home directory, and create or modify the line:

```
application/x-dvi; xhdvi %s
```

Your machine must already have the TEX essentials on board of course — in particular the pk font files, and the location of those font files needs to be communicated to the previewer. If XDVI is already working for you, XHDVI should work too. Details for getting XHDVI working on your machine are provided below.

For jumping to external documents from within the hypertexted DVI file, a couple of additional elements are needed, also desribed below for the case of XHDVI.

4.2 As an author

Here is where the power of TEX's macro capabilities appears. A working internal hypertext document can be made from a LATEX document with a one-line addition to the file, using Tanmoy Bhattacharya's hypertex macros. These macros convert the standard LATEX markup into hypertext links between the different sections of the document, so that references to equations, tables, footnotes, and section headings are in place, and bibliographic references and figures refer back at least to the bibliography entry or figure caption. These in turn may be set to refer to corresponding external documents but this process is not automatic — currently the author will have to add these references by hand, although automatic procedures can be envisioned. With an Internet connection, XHDVI can be used to preview the document and check that the references actually work, before the document is submitted to the archives.

The macros developed thus far use standard naming conventions for the underlying structures in LATEX and other standard macro packages, so that appending #equation.2.3, #page.7, #figure.4, #table.2, etc. to the URL for any TEX file processed with these packages will go to the right place, allowing easy hypertext reference to the internal structure of other documents.

In order to get started, however, you need to place these macro files in one of the standard areas that your TEX looks for input files (you can modify your TEXINPUTS environment variable to get it to look in your own directories). The needed macro files are itemized in the HyperTEX introductory document at
`http://xxx.lanl.gov/hypertex/index.html`
and can be obtained in one lump by anonymous ftp from
`ftp://snorri.chem.washington.edu/hypertex/hypermacros.Z`

4.3 As an e-print manager

Since we currently only have DVI previewers, an e-print server would have to serve the documents in preprocessed `.dvi` form. This means converting documents to HyperTEX if the author has not already done this, and possibly applying automated insertion of URL's corresponding to references in the bibliographic section. The manager could do this by hand but it might be rather time-consuming.

For ease of use, the best way to serve the documents is probably as a combined package of `.dvi` and `.ps` files that go together. This requires the e-print manager to create a new content-type associated with this package, and to supply an unpackaging program for the reader to place in their *mailcap* file, which automatically calls up XHDVI or another HyperTEX browser on the resultant main DVI file. The reason for doing this is that .ps files included by the standard *figures in TEX* macros will not generally be understood as remote documents, at least at the current level of previewer capabilities. Another option in this unpackaging method is to supply the TEX file itself, pipe it through a simple converter to HyperTEX and through TEX itself, and then call one of the HyperTEX viewers. These approaches are already in use at some locations (e.g. CERN).

When the pdf converter is available, the entire document should come as a single pdf file, unless the document refers to non-PostScript images or other inclusions in which case the packaging approach (or use of absolute URL's) remains necessary.

5 How do I get it?

Currently the following are available:

- XHDVI version 0.7d at
  `ftp://snorri.chem.washington.edu/hypertex/`
  A HyperTEX viewer based on `xdvi-18`, modified by Arthur Smith. Precompiled versions for various UNIX architectures are available in the same directory.
- `HyperTeXview.app` courtesy of Dmitri Linde (`dmitri@physics.stanford.edu` [also the author of INSTANTTEX.APP]) for NextStep, precompiled for Motorola and Intel-based NeXT machines. See
  `http://xxx.lanl.gov/hypertex/index.html`
  for availability.
- DVIHPS by Mark Doyle (`doyle@mmm.lanl.edu`) for conversion to a 'hyper' postscript that can be readily transformed to PDF.
- The macro and style files listed above by Tanmoy Bhattacharya, available at
  `ftp://nqcd.lanl.gov/people/tanmoy/hypertex/`

6 Details on xhdvi

XHDVI retains all the features of the latest version of XDVI (version 18) and adopts in addition many of the hypertext features of Mosaic, the most popular WWW browser. Hypertext links are underlined or altered in color (the underlining can be turned off) and a left-mouse click on a link causes the view to shift to the destination point for the link, as long as the destination is another DVI file. If the link is not to a DVI file, an external viewer is employed, following the mime and mailcap definitions or using standard defaults if those are not locally defined. A middle mouse click on a link brings up a new viewer whether or not the destination is a DVI file — this is intended to be useful to refer back to equations or to bring up footnotes, since the new DVI window is small. There are also a large number of keyboard accelerators, all described in detail in the man page.

In general, see the installation notes provided with XHDVI.

In outline what is needed is:

1. The compiled XHDVI program — precompiled binaries are available for Sun, NeXT, SGI, HP, IBM RS6000, or you can get the source and compile it yourself. Let me know of any compilation troubles — it's written in C.

2. The TEX fonts, at least in pk format. If DVI, DVIPS or some other DVI interpreter are working on your machine then they must be around somewhere. Otherwise you can get a small sample set of fonts from snorri.chem.washington.edu, or get the full TEX distribution from somewhere...
3. If the TEX fonts are not in the directory `/usr/local/lib/tex/fonts` you will either have to recompile XHDVI or use the environment variables, or you can add a symbolic link from `/usr/local/lib/text/fonts` to the right places. On a machine where you don't have superuser access and you have to install the TEX fonts yourself (a few Megabytes) you could put them in `~/TEX/fonts` and then set the environment variables:
   ```
   setenv XDVIFONTS $\sim$/TEX/fonts/pk
   setenv XDVIVFS   $\sim$/TEX/fonts/vf
   ```
4. Set up the connections between the Web browser and XHDVI. If you use mosaic for example,
   ```
   setenv WWWBROWSER /usr/local/bin/mosaic
   ```
   will let XHDVI know what to send HTML files to. To let mosaic know to bring up XHDVI for any DVI files, you need to have the line
   ```
   application/x-dvi; xhdvi %s
   ```
   in your `~.mailcap file`.
5. The application defaults file XHDVI should be installed in the standard application defaults directory on your machine, or you can take lines from it and modify them for your own taste and put them in your `~.Xdefaults` file. For example I use the following resource specifications to get a particular size and position of the window with white on black lettering and with the hyperlinks in cyan, and to remove the buttons:
   ```
   xhdvi*geometry: 800x600-0-0
   xhdvi*foreground: white
   xhdvi*background: black
   xhdvi*highlight: cyan
   xhdvi*expert: true
   ```
6. You need to have the ghostscript program on your machine and in your default execution path in order to view postscript from XHDVI. Similarly, other viewers defined in the mailcap file should be available on the machine.
7. You need to install the man page `xhdvi.man` in `/usr/local/man/man1` and add `/usr/local/man` to your `MANPATH` environment variable in order for *help* to work from XHDVI.

7 Some examples

This document is available in raw HyperLATEX format and in converted dvi format *via* anonymous ftp from
`ftp://snorri.chem.washington.edu/hypertex/hyper-intro.tex`.

The HyperTEX version of this paper uses the two-column APS journal style of revtex. The table of contents at the beginning is generated automatically with the LATEX `\tableofcontents` command.

See also the examples provided by Paul Ginsparg in the HyperTEX introductory document at
`http://xxx.lanl.gov/hypertex/index.html`.
Some of these are files randomly selected from the HEP archive, including LATEX, RevTEX, and other formats.

8 What still needs to be done?

Unfortunately, at this point reference to networked files (via URL's) suffers from a couple of problems. XHDVI does not yet include any of the network transport code that ordinary WWW browsers use, and the intention was to avoid having to add this layer of complexity by communications back and forth with a WWW browser. However, such communication is as yet not standardized, and suffers from its own problems. So currently, when XHDVI comes across a URL reference, it forwards it directly to the WWW browser (defined by environment or Xresource variables) so that a reference to an external DVI file would bring up a new instance of the WWW browser which would in turn bring up a new XHDVI viewer. This is a rather inelegant solution, but it is perhaps sufficient at the moment. A better solution will come along, and it may simply be inclusion of network transport code in the XHDVI viewer itself, to make it a competing WWW browser...

The other problem is that if brought up by a WWW browser, XHDVI is not provided with the absolute URL information used in obtaining the DVI file it is working on, and so cannot pass this information on to further instances. Therefore, relative URL's in a HyperTEX document (unless they can be guaranteed to be to local files that would have been transported along with the DVI file) will not work.

Both of the above are problems intrinsic to current WWW browsers, and we are working on promulgating solutions to these.

9 How do I stay in contact?

Bug reports or other problems should be sent to me by e-mail (`asmith@mammoth.chem.washington.edu`). The hypertex-dev mailing list is available for those interested in assisting the development and implementation of HyperTEX software. If you merely want to stay informed about developments of new HyperTEX software, you can join the hypertex-announce mailing list. Subscriptions to both are handled automatically through e-mail to:

`majordomo@snorri.chem.washington.edu`

The Hyperlatex Markup Language

Otfried Schwarzkopf

Vakgroep Informatica, Universiteit Utrecht,
Postbus 80.089, 3508 TB Utrecht, the Netherlands
otfried@cs.ruu.nl

Januari 31, 1995

1 Introduction

Hyperlatex is a little package that allows you to use LaTeX to prepare documents in HTML (the *hypertext markup language* used by the *world wide web*), and, at the same time, to produce a fine printed document from your input. You can use all of LaTeX's power for the printed output, and you don't have to learn a new language for creating hypertext documents.

Using Hyperlatex is easy. You create a file *document.tex*, say, containing LaTeX commands — the same commands you are used to — plus a few additional directives controlling conversion to HTML. If you use the command

```
latex document.tex
```

then your file will be processed by LaTeX, resulting in a DVI-file, which you can print as usual.

On the other hand, you can run the command

```
hyperlatex document.tex
```

and your document will be converted to HTML format, presumably to a set of files called *document.html*, *document_1.html*, ... (These files are created in a separate directory which you can specify within the source file using the `\htmldirectory` command.) You can then use any HTML-viewer or WWW-browser, such as `Mosaic`, to view the document.

This document describes how to use the Hyperlatex package. It tells you the *mechanics* of setting up an input file for LaTeX and HTML, and discusses the subset of LaTeX commands which are understood and converted to HTML tags by the `hyperlatex` converter. This manual does not explain *what* to write in a WWW-document. There are style guides available, which you might want to consult. Writing an on-line document is not the same as writing a paper. I hope that Hyperlatex will help you to do both properly.

We assume that you are familiar with LaTeX, and that you have at least some familiarity with hypertext documents — that is, that you know how to use one of the WWW-browsers and understand what a *hyperlink* is.

If you want, you can have a look at the TeX source. You can use it as a template in writing your own documents, and illustrates some points discussed here.

While writing and testing Hyperlatex, I have converted several LaTeX documents into Hyperlatex format. It turns out that it takes only a few minutes for a document that does not use much mathematics or that defines lots of its own commands. One example I used was *A few rules from 'A Handbook for Scholars'* by Mark de Berg. A big document written with Hyperlatex is the *Ipe Manual*, which has about 50 pages in the printed version and 70 nodes as a HTML-document. Others at our department have used Hyperlatex, for instance to put the department's study guide (more than 200 nodes) on the world wide web.

If you have used Hyperlatex to make some document available on the world wide web, I would be happy to hear about it. I would certainly like to set up a list with demo documents.

A final footnote: The converter to HTML implemented in Hyperlatex is written in GNU Emacs Lisp. You can use it directly from Emacs. But even if you don't use Emacs, even if you don't like Emacs, or even if you subscribe to `alt.religion.emacs.haters`, you can happily use Hyperlatex. Hyperlatex can be invoked from the shell (as shown above), and you will never know that Emacs is responsible for the finely formatted document which you get.

The Hyperlatex code is based on the Emacs Lisp macros of the `latexinfo` package.

2 Using Hyperlatex

Using Hyperlatex is easy. You write your document in a LaTeX-file *document.tex*, using a certain subset of LaTeX commands, and some additional commands that control the conversion to HTML, and to make hyperlinks between parts of your document.

To make a printed document, you then run LaTeX on your file, like usual, and follow the standard procedures for printing the DVI-file.

To turn your document into HTML format, you can either run the `hyperlatex` shell script, which invokes Emacs and runs the conversion macros, or you can do conversion directly from Emacs. The `hyperlatex` script takes the following arguments:

```
hyperlatex files
```

You have to specify the full filenames, including the extension *.tex*.

To run conversion from within Emacs, put the following line in you *.emacs* file:

```
(autoload 'hyperlatex-format-buffer
                                "hyperlatex1")
```

Then you can call `hyperlatex-format-buffer` in the buffer containing the LaTeX input file. But note that the shell script version produces better error messages.

A typical HTML document consists of a set of files. In HTML-speak these files are also called 'documents'. In this manual we take the LaTeX point of view, and call 'document' what is enclosed in a `document` environment. We will use the term *node* for the individual files of the HTML document.

The node files are created in a directory which you have to specify in the preamble of your source file. You also have to specify the *base name* of the HTML-document:

```
\htmldirectory{directory}
\htmlname{basename}
```

The actual files created by `hyperlatex` are called *directory/basename.html*, *directory/basename_1.html*, *directory/basename_2.html*, and so on. The filename can be changed for individual nodes using the `\xname` command.

The entry point for your document will be the file *directory/basename.html*. This means that you can view your HTML-document using `Mosaic` as follows.

```
Mosaic directory/basename.html
```

3 Controlling the conversion to Html

Hyperlatex automatically partitions the document into several nodes. This is done based on the LaTeX sectioning. The section commands `\chapter`, `\section`, `\subsection`, `\subsubsection`, `\paragraph`, and `\subparagraph` are assigned levels 1 to 6. (If you use the *article* document style, `\section` to `\subparagraph` are given levels 1 to 5, as there are no chapters).

The `\htmldepth` command in the preamble determines at what depth separate nodes are created. The default setting is 4, which means that (for *article* style) sections, subsections, and subsubsections are given their own nodes, while paragraphs and subparagraphs are put into the node of their parent subsection. You can change this by putting

```
\htmldepth{depth}
```

in the preamble. A value of 1 means that the full document will be stored in a single file.

A HTML file needs a *title*. This *must be set* in the preamble using the `\htmltitle` command. Use something short but helpful. The title you specify is used directly for the top node of your document. The other nodes get a title composed of this and the section heading.

It is common practice to put a short notice at the end of every HTML node, giving a reference to the author. This can be done by using the `\htmladdress` command in the preamble.

The individual nodes of a HTML document are linked together using *hyperlinks*. Hyperlatex automatically places buttons on every node that link it to the previous and next node of the same depth, if they exist, and a button to go to the parent node.

Furthermore, Hyperlatex automatically adds a menu to every node, containing pointers to all subsections of this section. (Here, 'section' is used as the generic term for chapters, sections, subsections, ...) This may not always be what you want. You might want to add nicer menus, with a short description of the subsections. In that case you can turn off the automatic menus by putting

```
\htmlautomenu{0}
```

in the preamble. On the other hand, you might also want to have more detailed menus, containing not only pointers to the direct subsections, but also to all subsubsections and so on. This can be achieved by putting

```
\htmlautomenu{depth}
```

in the preamble, where *depth* is the desired depth of recursion. The default behavior corresponds to a *depth* of 1.

A final note: All these commands must start at the beginning of a line, if you want Hyperlatex to see them.

4 Parsing by LaTeX and Hyperlatex

You are writing an input file that has to be read by LaTeX as well as the Hyperlatex converter. The parsing done by LaTeX is complex, and has many of us surprised in certain situations. It was hopeless to try to imitate this complex behavior using a modest collection of Emacs Lisp macros. Nevertheless, Hyperlatex should behave well on your LaTeX files. If your source is comprehensible to LaTeX (with the *hyperlatex.sty* package), then Hyperlatex should not have *syntactical* problems with it. There is, however, one difference in parsing arguments: In LaTeX, you can write

```
\emph example,
```

and what you will get is '*e*xample'. Hyperlatex will complain about this. To get the same effect, you will have to write

```
\emph{e}xample.
```

Hyperlatex has been designed to understand a certain subset of LaTeX. It will treat all other LaTeX instructions with an error message. This does not mean that you should not use any of these instructions for getting exactly the printed document that you want. By all means, do. However, I felt it was safer if Hyperlatex did not ignore any commands it doesn't know. So you will have to hide those commands from Hyperlatex using the escape mechanism.

Here is what your input file should roughly look like:

```
\documentclass{article}
\usepackage{hyperlatex}

\htmldirectory{HTML directory}
\htmlname{base filename of HTML nodes}
```

```
\htmltitle{title of HTML nodes}
\htmladdress{otfried@cs.ruu.nl}

.... more LaTeX declarations, if you want

\title{Title for LaTeX document}
\author{Author for LaTeX document}

\begin{document}

\maketitle
\section{Introduction}

\topnode{Welcome to this HTML Document}

This is the beginning of the section
titled 'Introduction' in the printed
manual, and at the same time the
beginning of the top node of the HTML
document....
```

If you are still using LaTeX2.09, replace the first two lines by

```
\documentstyle[hyperlatex]{article}
```

Note the use of the *hyperlatex* package. It contains the definitions for some LaTeX extensions useful in Hyperlatex, and also turns on the special input mode.

For the HTML document, Hyperlatex ignores everything before the line starting with `\topnode` (there may only be white space on this line before the command). The `\topnode` command specifies the heading for the *top node* of the HTML document. It does not produce any output in the printed manual.

5 A LaTeX subset — Getting started

Starting with this section, we take a stroll through the LaTeX-book [1], explaining all features that Hyperlatex understands, additional features of Hyperlatex, and some missing features. For the LaTeX output the general rule is that *no LaTeX command has been changed*. If a familiar LaTeX command is listed in this manual, it is understood both by LaTeX and the Hyperlatex converter, and its LaTeX meaning is the familiar one. If it is not listed here, you can still use it by escaping into TeX-only mode, but it will then have effect in the printed manual only.

5.1 Hyperlatex input mode

There are ten characters that LaTeX treats as special characters, which means that they do not simply typeset themselves:

```
# $ % & ~ _ ^ \ { }
```

Hyperlatex has only five special characters:

```
\ { } % ~
```

The remaining five characters are not special in Hyperlatex. They simply typeset themselves. To typeset one of the special characters, use

```
\=   \{   \}  \%  \sim
```

Note that `\{`, `\}`, and `\sim` exist in LaTeX, but only work in math mode. These, and the two shortcuts `\=` and `\+` are actually the only LaTeX commands whose definition have been changed. You can use `\back` as a synonym for `\=`, to typeset a backslash.

If you need the special meaning of one of LaTeX's special characters, you need to use an escape to LaTeX. The Hyperlatex input mode is turned on by `\begin{document}`. This means that you can still use the regular LaTeX special characters in the preamble. (For technical reasons the special input mode is turned on by `\topnode` if you are using LaTeX2.09.)

We said above that the remaining characters typeset themselves. This has to be taken with a grain of salt. LaTeX still obeys ligatures, which turns `ffi` into 'ffi', and some characters, like `>`, do not resemble themselves in some fonts (`>` looks like > in roman font). The only characters for which this is critical are `<`, `>`, and `|`. Better use them in a typewriter-font (this includes the `example` and `verbatim` environments and the `\code` and `\kbd` fonts). Note that `?‘` is a ligature even in `\typew` font and displays as `?‘`, but displays correct in the other (logical) fonts listed above.

Like LaTeX, the Hyperlatex converter understands that an empty line indicates a new paragraph. You can achieve the same effect using the command `\par`.

5.2 Dashes and Quotation marks

Hyperlatex translates a sequence of three dashes `---` into two dashes `--`. The quotation mark sequences `''` and `‘‘` are translated into simple quotation marks `"`.

5.3 Simple text generating commands

The following simple LaTeX macros are implemented in Hyperlatex:

- `\LaTeX` produces LaTeX.
- `\TeX` produces TeX.
- `\LaTeXe` produces LaTeX2$_\varepsilon$.
- `\copyright` produces ©.
- `\ldots` produces three dots . . .
- `\minus` produces a minus sign −.
- `\quad` and `\qquad` produce some empty space.
- `\ss` produces ß.
- `\today` produces 17th May 1995— although this might depend on when you use it . . .

5.4 Emphasizing Text

Hyperlatex understands the following physical font specifications of LaTeX2$_\varepsilon$:

- `\textbf` for **bold**
- `\textit` for *italic*
- `\textsc` for SMALL CAPS
- `\texttt` for `typewriter`
- `\underline` for underline

Note that these font changes are properly cumulative in LaTeX2$_\varepsilon$ and in the `netscape` browser, but are not in LaTeX2.09 and in older HTML browsers. The following commands are supported for backwards compatibility:

- \bf and \bold for **bold**
- \it and \italic for *italic*
- \scap for SMALL CAPS
- \typew for `typewriter`

So you can write

```
{\it italic text}
```

but also

```
\textit{italic text}
```

Note that if you use the old LaTeX versions \it and \bf, the command must come directly after an opening brace. You may **not** write

```
{roman text \it italic text}
```

The HTML guidelines encourage you to think in terms of *logical concepts* instead of physical fonts. So, do not write \textit{filename}, but write \file{filename}. This has the advantage that the reader of the document can still decide how she wants the logical concept 'filename' to be rendered (for instance in a light green cyrillic font, if she wants). Here are the logical fonts available in HTML:

- \cit for *citations*.
- \code for `code`.
- \dfn for *definitions*.
- \em and \emph for *emphasized text*.
- \file for *filenames*.
- \kbd for `keyboard input`.
- \samp for `sample input`.
- \strong for **strong emphasis**.
- \var for *variables*.

Finally, you can use 20\dmn{in} to typeset the dimension 20 in.

You can use \/ to separate slanted and non-slanted fonts (it will be ignored in the HTML-version).

5.5 Changing Type Size

LaTeX's declarations for changing the type size are all understood, and HTML-tags are generated for them. Note, however, that currently only the `netscape` browser interprets these size changes, other browsers will simply ignore them. The commands are \tiny, \scriptsize, \footnotesize, \small, \normalsize, \large, \Large, \LARGE, \huge, and \Huge. As the \it command, these commands have to immediately follow an opening brace. So you can write

```
{\large larger text},
```

but you may **not** write

```
{normal text \large larger text}
```

In the HTML version, these font sizes are relative to the node's basefont size (\normalsize being the basefont size, \large being the basefont size plus one etc.) To set a node's basefont size, you can use the command

```
\html{basefont size=x}
```

where *x* is a number between 1 and 7.

5.6 Preventing line breaks

The ~ is a special character in Hyperlatex, and is replaced by a HTML tag for 'non-breakable space'. It seems, however, that the current `Mosaic` version does not honor this, and simply treats it as a space. Nevertheless, ~'s are useful for the printed document.

5.7 Footnotes

are not yet implemented.

5.8 Formulas

There is no *math mode* in HTML, and all commands related to it are rejected. It is probably useless to try to convert a paper with a lot of mathematics into HTML format anyway.

However, sometimes you want to include simple expressions like 'the segment from point p to point q' or 'Pythagoras' theorem states that $a^2 + b^2 = c^2$.' In such cases you would like to have the properly formatted version of the formula in the printed document, and some approximation in the HTML-version. This can be done with the new \math command:

```
\math{argument}
```

In LaTeX, this command typesets the *argument*, which is read in *math mode* with all special characters enabled. Hyperlatex simply typesets the *argument* without any special treatment (but embedded commands are expanded). Often the LaTeX math expression does not look good when put into the HTML-document untreated, contains unknown commands, or you simply want something different. You might, for instance, want to typeset the ith element (`the \math{i}th element`) of an array as a_i in LaTeX, but as `a[i]` in HTML. This can be done with the optional argument of \math:

```
\math[HTML-version]{LaTeX-version}
```

In this example: \math[\code{a[i]}]{a_{i}}.

As mentioned, there is no math mode in HTML and you have to do with an approximation of the formula. Nevertheless, if you want, you can still have them displayed in an italic font. To do so, place a \htmlmathitalics command in your preamble. It must start on the first character of a line.

5.9 Ignoring input

The percent character % introduces a comment in Hyperlatex. Alternatively, you can use the command \C. Everything after a % or \C up to the end of the line is ignored, as well as any white space on the beginning of the next line.

5.10 Document class and title page

This material appears before the `\topnode` command and is therefore ignored by the Hyperlatex converter. You can use everything you want there.

5.11 Sectioning

The sectioning commands `\section`, `\subsection`, `\subsubsection`, `\paragraph`, and `\subparagraph` are recognized by Hyperlatex and used to partition the document in nodes. The `\chapter` command is recognized if the document class is not `article`. You can also use the starred version and the optional argument for the sectioning commands. The star will be ignored, as Hyperlatex does not number sections, and the optional argument will be used for node titles and in menus generated by `htmlmenu`.

You can use `\protect`. It will be ignored in the HTML-version.

5.12 Displayed material

The `quote`, `quotation`, and `verse` environment are all implemented by the Hyperlatex converter — but they are all identical!

The `center` environment is realized using an HTML tag that is currently only understood by the `netscape` browser.

To make lists, you can use the `itemize`, `enumerate`, and `description` environments. You *cannot* specify an optional argument to `\item` in `itemize` or `enumerate`, and you *must* specify one for `description`.

All these environments can be nested.

The `\\` command is recognized, with and without `*`.

There is also a `menu` environment, which looks like an `itemize` environment, but is somewhat denser since the space between items has been reduced. It is only meant for single-line items.

6 Conditional Compilation: Escaping into one mode

In many situations you want to achieve slightly (or maybe even totally) different behavior of the LaTeX code and the HTML-output. Hyperlatex offers several different ways of letting your document depend on the mode.

6.1 LaTeX versus HTML mode

The easiest way to put a command or text in your document that is only included in one of the two output modes it by using a `\texonly` or `\htmlonly` command. They ignore their argument, if in the wrong mode, and otherwise simply expand it:

```
We are now in
       \texonly{\LaTeX}\htmlonly{HTML}-mode.
```

Another possibility is by prefixing a line with `\T` or `\H`. `\T` is equal to `\C` in HTML-mode, and a noop in LaTeX-mode, and for `\H` it is the other way round:

```
We are now in
\T \LaTeX-mode.
\H HTML-mode.
```

The last way of achieving this effect is useful when there are big chunks of text that you want to skip in one mode — a HTML-document might skip a section with a detailed mathematical analysis, a LaTeX-document will not contain a node with lots of hyperlinks to other documents. This can be done using the `iftex` and `ifhtml` environments:

```
We are now in
\begin{iftex}
  \LaTeX-mode.
\end{iftex}
\begin{ifhtml}
  HTML-mode.
\end{ifhtml}
```

6.2 Escaping to 'real' LaTeX

Even within the `iftex` environment the special input mode of Hyperlatex is still effective. Sometimes you will want to be able to use the full power of LaTeX with all its special characters. This can be done in a `tex` environment. It is equivalent to `iftex`, but also turns on the five special characters that make the difference between 'real' TeX and Hyperlatex.

Here is another neat construction that lets you go into 'real' TeX mode for a single line:

```
\T {\tex ... and now we are in real TeX
                                    mode ...  }
```

The `\T` command escapes from Hyperlatex, and the `\tex` command sets TeX's special characters.

6.3 Flags — more on conditional compilation

You can also have sections of your document that are included depending on a flag which you have set or cleared before. To set a flag, use

```
\set{flag}
```

To clear a flag, use

```
\clear{flag}
```

Both commands can be used both in the preamble and in the body of the document. If used in the preamble, they must start at the beginning of the line or else be prefixed with `\H` and whitespace, if the Hyperlatex converted has to see them.

Then you can include parts of your document based on some flag:

```
\begin{ifset}{flag}
  Flag flag is set!
\end{ifset}

\begin{ifclear}{flag}
  Flag flag is not set!
\end{ifset}
```

You can set and clear a flag more than once. It is not an error to test a flag which has not been defined with `\set` or `\clear`. It is considered cleared.

7 Carrying on

In this section we continue to Chapter 3 of the LaTeX-book, dealing with more advanced topics.

7.1 Accents

Hyperlatex recognizes the accent commands

```
\'   \`   \^   \~
```

However, not all possible accents are available in HTML. Hyperlatex will make a HTML-entity for the accents in ISO Latin 1, but will reject all other accent sequences. The command `\c` can be used to put a cedilla on a letter 'c' (either case), but on no other letter. The following is legal

```
Der K{\"o}nig sa\ss{} am wei{\ss}en Strand
von Cura\c{c}ao und nippte an einer
Pi\~{n}a Colada \ldots
```

and produces

> Der König saß am weißen Strand von Curaçao und nippte an einer Piña Colada . . .

Not legal are `Ji{\v r}\'{\i}`, or `Erd\H{o}s`. To get a 'í', you have to type `\'{\i}`, not `\'\i`.

7.2 Defining commands and environments

Hyperlatex understands the simplest type of command definitions, namely commands without parameters, and *only in the preamble*. The `\newcommand` command must start at the beginning of the line, or must be prefixed by a `\H` command and white space. The same holds for new environments. Here are some legal examples:

```
\newcommand{\Html}{\scap{html}}

\T\newcommand{\bad}{$\surd$}
\H\newcommand{\bad}{\htmlimage{%
                        badexample_bitmap.xbm}}

\newenvironment{badexample}{\begin{description}
                \item[\bad]}{\end{description}}

\newcommand{\ipe}{\italic{Ipe}}

\H \newenvironment{smallexample}{%
              \begin{example}}{ \end{example}}
\T \newenvironment{smallexample}{\begingroup
                    \small\begin{example}}{%
                      \end{example}\endgroup}
```

The `\bad` command and the `smallexample` environments are good examples for conditional compilation. The `smallexample` environment is equal to `example` in HTML, but is typeset in a smaller font in the LaTeX document.

It is possible to trick Hyperlatex into defining a new command with an argument, if the HTML-implementation of the new command simply typesets the argument:

```
\T \newcommand{\frameit}[1]{\fbox{#1}}
\H \newcommand{\frameit}{\italic}
```

The new command `\frameit` will typeset its argument in italics in HTML-mode, but will put a frame around it in LaTeX.

There is no `\renewcommand`. You cannot redefine any predefined commands.

7.3 Theorems and such

There is no `\newtheorem` command. But you can define an environment which does approximately the same:

```
% LaTeX definition
\newtheorem{guess}{Conjecture}

% HTML definition
\H \newenvironment{guess}{\begin{quotation}%
                          \bold{Conjecture.}
   \html{I}}{\html{/I}\end{quotation}}
```

(The `\html` command generates plain HTML-tags. The 'I' and '/I' tags used here turn on and off italics mode.)

7.4 Figures and other floating bodies

You can use `figure` and `table` environments and the `\caption` command. They will not float, but will simply appear at the position in the text. No special space is left around them, so put a `center` environment in a figure. The `table` environment is mainly used with the `tabular` environment below.

7.5 Lining it up in columns

There is a weak implementation of the `tabular` environment available in Hyperlatex. First of all, the &-character is not special in Hyperlatex, so instead you have to use the `\S` command to separate columns.

To produce the HTML-version of the table, Hyperlatex removes all the `\S` commands *with any following white space* and the `\\` or `\\*` commands. The result is not formatted any more, and simply included in the HTML-document as a 'preformatted' display. This means that if you format your source file properly, you will get a well-formatted table in the HTML-document — but it is fully your own responsibility.

You can also use the `\hline` command to include a horizontal rule. Here is an example:

```
\begin{table}[htp]
\caption{Keyboard shortcuts for \ipe{}}
\begin{center}
\begin{tabular}{|l|lll|}
\hline
~      \S Left Mouse \S Middle Mouse\S Right Mouse \\
\hline
Plain \S (start drawing)\S move   \S select       \\
Shift \S scale          \S pan    \S select more  \\
Ctrl  \S stretch        \S rotate \S select type  \\
Shift+Ctrl\S ~          \S ~   \S select more type\\
\hline
\end{tabular}
\end{center}
\end{table}
```

Note the use of the ~-character. Without it the `\hline` command would eat up space up to the next `\S` command, and the same holds for the two `\S` commands on the last line. The example is typeset as follows:

Table 1: *Keyboard shortcuts for Ipe*

	Left Mouse	Middle Mouse	Right Mouse
Plain	(start drawing)	move	select
Shift	scale	pan	select more
Ctrl	stretch	rotate	select type
Shift+Ctrl			select more type

7.6 Simulating typed text

The `verbatim` environment and the `\verb` command are implemented. The starred varieties are currently not implemented. (The implementation of the `verbatim` environment is not the standard LaTeX implementation, but the one from the *verbatim.sty* style by Rainer Schöpf). The command `\+`*verb*+ can be used as a shortcut for `\verb+`*verb*+.

Furthermore, there is another, new environment `example`. `example` is also useful for including program listings or code examples. Like `verbatim`, it is typeset in a typewriter font with a fixed character pitch, and obeys spaces and line breaks. But here ends the similarity, since `example` does *not* turn off the five special characters. Using this you can still use font changes within an `example` environment, and you can also place hyperlinks there. Here is an example:

```
To clear a flag, use
\begin{example}
  \+\clear{+\var{flag}\}
\end{example}
```

Note also that an `example` environment is indented automatically, while a `verbatim` environment is not. In the LaTeX document, you can set the amount of indentation by setting `\exampleindent`:

```
\setlength{\exampleindent}{4mm}
```

8 Moving information around

In this section we deal with questions related to cross referencing between parts of your document, and between your document and the outside world. Here lie some of the big differences between a printed paper and a HTML-document. Where you would have an expression such as 'More details can be found in the classical analysis by Harakiri [8]' in the printed paper, the HTML-document would include a hyperlink to Harakiri's work.

8.1 Cross-references

You can use the `\label{`*label*`}` command to attach a *label* to a position in your document. This label can be used to create a hyperlink to this position from any other point in the document. This is done using the `\link` command:

```
\link{anchor}{label}
```

This command typesets anchor, expanding any commands in there, and makes it an active hyperlink to the position marked with *label*:

```
This parameter can be set in the
\link{configuration panel}{sect:con-panel}
                          to influence ...
```

The `\link` command does not do anything exciting in the printed document. It simply typesets the text *anchor*. If you also want a reference in the LaTeX output, you will have to add a reference using `\ref` or `\pageref`. This reference has to be escaped from the Hyperlatex converter. Sometimes you will want to place the reference directly behind the *anchor* text. In that case you can use the optional argument to `\link`:

```
This parameter can be set in the
\link{configuration
  panel}[~(Section~\ref{sect:con-panel})]%
          {sect:con-panel} to influence ...
```

The optional argument is ignored in the HTML-output. In most cases, you will need a `\ref`erence to the label already given in the `\link` command. To save you same typing, the `\link` command therefore defines `\Ref` and `\Pageref` (with capitals) to be `\ref{`*label*`}` and `\pageref{`*label*`}`, where *label* is the label used in the `\link` command. These definitions are already active when the optional argument is expanded. This means that we can rewrite the example above as:

```
This parameter can be set in the
\link{configuration panel}[~(Section~\Ref)]%
            {sect:con-panel} to influence ...
```

Often this format is not useful, because you want to put it differently in the printed manual. Still, as long as the reference comes after the `\link` command, you can use `\Ref` and `\Pageref`.

```
After \link{setting the parameter}{%
      sect:con-panel} it is not difficult
      to show that the dependence of the ....
      .... is obvious\texonly{ (see also
      Section~\Ref)}.
```

Note that when you use LaTeX's `\ref` command, the label does not mark a *position* in the document, but a certain *object*, like a section, equation etc. It sometimes requires some care to make sure that both the hyperlink and the printed reference point to the right place, and sometimes you will have to place two labels. The HTML-label tends to be placed *before* the interesting object — a figure, say — , while the LaTeX-label tends to be put *after* the object (when the `\caption` command has set the counter for the label).

A special case occurs for section headings. Always place labels *after* the heading. In that way, the LaTeX reference will be correct, and the Hyperlatex converter makes sure that the link will actually lead to a point directly before the heading — so you can see the heading when you follow the link.

8.2 Links to external information

You can place a hyperlink to a given URL (Universal Resource Locator) using the `\xlink` command. Like the `\link` command, it takes an optional argument, which is typeset in the printed output only:

```
\xlink{anchor}{URL}
\xlink{anchor}[printed reference]{URL}
```

In the HTML-document, *anchor* will be an active hyperlink to the object *URL*. In the printed document, *anchor* will simply be typeset, followed by the optional argument, if present.

8.3 Links into your document

The Hyperlatex converter automatically partitions your document into HTML-nodes and generates HTML-tags for your `\label`'s. These automatically created names are simply numbers, and are not useful for external references into your document — after all, the exact numbers are going to change whenever you add or delete a section or label, or when you change the `\htmldepth`.

If you want to allow links from the outside world into your new document, you will have to do two things: First, you should give that HTML node a mnemonic name that is not going to change when the document is revised. Furthermore, you may want to place a mnemonic label inside the node.

The `\xname{`*name*`}` command is used to give the mnemonic name *name* to the *next* node created by Hyperlatex. This means that you ought to place it *in front of* a sectioning command. The `\xname` command has no function for the LaTeX-document. No warning is created if no new node is started in between two `\xname` commands.

If you need an HTML label within a node to be referenced from the outside, you can use the `\xlabel{`*label*`}` command. *label* has to be a legal HTML label.

Here is an example: The section 'Changes between Hyperlatex 1.0 and Hyperlatex 1.1' in this document starts as follows.

```
\xname{hyperlatex_changes}
\section{Changes from from Hyperlatex~1.0 to
                            Hyperlatex~1.1}
\label{sec:changes}
```

It can be referenced inside this document with `\link{Changes}{sec:changes}`, and both inside and outside this document with `\xlink{Changes}{hyperlatex_changes.html}`.

The entry about `\xname` and `\xlabel` in that section has been marked using `\xlabel{external_labels}`. You can therefore directly refer to that position from anywhere using

```
\xlink{xlabel is new}{%
   hyperlatex_changes.html#external_labels}
```

8.4 Bibliography and citation

Hyperlatex understands the `thebibliography` environment. Like LaTeX, it creates a section titled 'References'. The `\bibitem` command is equivalent to `\par`, and sets a label with the given *cite key* at the given position. This means that you can use the `\link` command to define a hyperlink to a bibliography entry. The command `\Cite` is defined analogously to `\Ref` and `\Pageref` by `\link`. If you define a bibliography like this

```
\begin{thebibliography}{99}
   \bibitem{latex-book}
   Leslie Lamport, \cit{\LaTeX: A Document
                        Preparation System,}
   Addison-Wesley, 1986.
\end{thebibliography}
```

then you can add a reference to the LaTeX-book as follows:

```
... we take a stroll through the
\link{\LaTeX-book}[~\Cite]{latex-book},
                                explaining ...
```

Hyperlatex also understands the `\bibliographystyle` command (which is ignored) and the `\bibliography` command. It reads the *.bbl* file, inserts its contents at the given position and proceeds as usual. Using this feature, you can include bibliographies created with BibTeX in your HTML-document! It would be possible to design a WWW-server that takes queries into a BibTeX database, runs BibTeX and Hyperlatex to format the output, and sends back a HTML-document.

8.5 Splitting your input

The `\input` command is implemented in Hyperlatex. The subfile is inserted into the main document, and typesetting proceeds as usual. You have to include the argument to `\input` in braces.

8.6 Making an index or glossary

The Hyperlatex converter understands the commands `\index` and `\cindex`, which are synonymous. It collects the entries specified with these commands, and you can include a sorted index using `\htmlprintindex`. This index takes the form of a menu with hyperlinks to the positions where the original `\index` commands where located. You can specify a different sort key for an index intry using the optional argument of `\cindex`:

```
\cindex[index]{\verb+\index+}
```

This entry will be sorted like '`index`', but typeset in the index as '`\verb+\index+`'.

The *hyperlatex.sty* style defines `\cindex` as follows:

- `\cindex{`*entry*`}` is expanded to `\index{`*entry*`}`, and
- `\cindex[`*sortkey*`]{`*entry*`}` ist expanded to `\index{`*sortkey*`@`*entry*`}`.

This realizes the same behavior as in the Hyperlatex converter if you use the index processor `makeindex`. If not, you will have to consult your *Local Guide* and redefine `\cindex` appropriately.

The index in this manual was created using `\cindex` commands in the source file, the index processor `makeindex` and the following code:

```
\H \section*{Index}
\H \htmlprintindex
\T \input{hyperlatex.ind}
```

9 Designing it yourself

In this section we discuss the commands used to make things that only occur in HTML-documents, not in printed papers. Practically all commands discussed here start with `\html`, indicating that the command has no effect whatsoever in LaTeX.

9.1 Making menus

The `\htmlmenu` command generates a menu for the subsections of the current section. It takes a single argument, the depth of the desired menu. If you use `\htmlmenu{2}` in a subsection, say, you will get a menu of all subsubsections and paragraphs of this subsection.

If you use this command in a section, no automatic menu for this section is created.

A typical application of this command is to put a 'master menu' in the top node, containing all sections of all levels of the document. This can be achieved by putting `\htmlmenu{6}` in the text for the top node.

9.2 Rulers and images

The command `\htmlrule` creates a horizontal rule spanning the full screen width at the current position in the HTML-document. It has an optional argument that you can use to add the additional tags `size`, `width`, `align`, and `noshade`. These additional tags are currently only understood by the `netscape` browser. Here is an example.

```
\htmlrule[width=70% align=center]
```

[width=70

The command `\htmlimage{`*URL*`}` makes an inline bitmap with the given *URL*. It takes an optional argument that can be used to specify the additional tags understood by some HTML browsers. One of the letters 't', 'c', 'b', 'l', or 'r' can be specified as a shortcut for the alignments 'top', 'center', 'bottom', 'left', or 'right'. So `\htmlimage[c]{image.xbm}` includes the image in *image.xbm*, vertically centered at the current text position. A more complicated example is

```
\htmlimage[align=left width=50 height=75
                  hspace=3]{image.jpg}
```

(Note that *jpeg* inlined images are currently only understood by the `netscape` browser.)

This is what I use for figures in the Ipe Manual that appear in both the printed document and the HTML-document:

```
\begin{figure}
  \caption{The Ipe window}
  \begin{center}
    \T {\tex\Ipe{window.ipe}}
    \H \htmlimage{window.gif}
  \end{center}
\end{figure}
```

(`\Ipe` is the command to include 'Ipe' figures. Since the figure contains math mode material, it has to be escaped using `\tex`.)

9.3 Adding raw HTML

Hyperlatex provides two commands to access the HTML-tag level.

`\html{`*tag*`}` creates the HTML tag `<tag>`, and `\htmlsym{`*entity*`}` creates the HTML entity description `&entity;`.

The `\htmlsym` command is useful if you need symbols from the ISO Latin 1 alphabet which are not predefined in Hyperlatex. You can, for instance, define the ligature `\AE` as in TEX using

```
\H \newcommand{\AE}{\htmlsym{AElig}}
```

9.4 Turning TEX into bitmaps

There can be many things in a LATEX-file that Hyperlatex doesn't understand: equations, fancy tables, picture environments — the list is endless. Especially equations appear quite often and are pretty hard to represent in HTML. Sometimes the only sensible way to incorporate them into a HTML-document is by turning them into a bitmap. Hyperlatex has an environment `gif` that does exactly this: In the HTML-version, it is turned into a reference to an inline bitmap (just like `\htmlimage`). In the LATEX-version, the `gif` environment is equivalent to a `tex` environment. Note that running the Hyperlatex converter doesn't create the bitmaps yet, you have to do that in an extra step as described below.

The `gif` environment has three optional and one required arguments:

```
\begin{gif}[tags][resolution][                    fon
  TEX material ...
\end{gif}
```

For the LATEX-document, this is equivalent to

```
\begin{tex}
  TEX material ...
\end{tex}
```

For the HTML-version, it is equivalent to

```
\htmlimage[tags]{name.gif}
```

The other two parameters, *resolution* and *font_resolution*, are used when creating the `gif`-file. They default to 100 and 300 dots per inch.

Here is an example:

```
\htmlonly{\par}
\begin{gif}{eqn1}
  \[
  \sum_{i=1}^{n} x_{i} = \int_{0}^{1} f
  \]
\end{gif}
```

produces the following output:

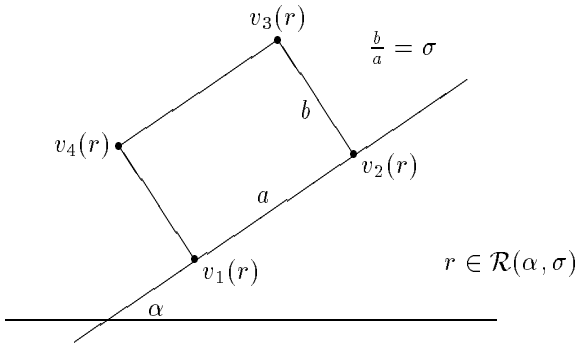
$$\sum_{i=1}^{n} x_i = \int_0^1 f$$

We could as well include a picture environment. The code

```
\begin{center}
 \begin{gif}[b][80]{boxes}
  \setlength{\unitlength}{0.1mm}
  \begin{picture}(700,500)
   \put(40,-30){\line(3,2){520}}
   \put(-50,0){\line(1,0){650}}
   \put(150,5){\makebox(0,0)[b]{$\alpha$}}
   \put(200,80){\circle*{10}}
   \put(210,80){\makebox(0,0)[lt]{$v_{1}(r)$}}
   \put(410,220){\circle*{10}}
   \put(420,220){\makebox(0,0)[lt]{$v_{2}(r)$}}
   \put(300,155){\makebox(0,0)[rb]{$a$}}
   \put(200,80){\line(-2,3){100}}
   \put(100,230){\circle*{10}}
```

```
    \put(100,230){\line(3,2){210}}
    \put(90,230){\makebox(0,0)[r]{$v_{4}(r)$}}
    \put(410,220){\line(-2,3){100}}
    \put(310,370){\circle*{10}}
    \put(355,290){\makebox(0,0)[rt]{$b$}}
    \put(310,390){\makebox(0,0)[b]{$v_{3}(r)$}}
    \put(430,360){\makebox(0,0)[l]{$\frac{b}{a} =
                                        \sigma$}}
    \put(530,75){\makebox(0,0)[l]{$r \in
                        {\cal R}(\alpha, \sigma)$}}
  \end{picture}
 \end{gif}
\end{center}
```

creates the following image.



It remains to describe how you actually generate those bitmaps from your Hyperlatex source. This is done by running LaTeX on the input file, setting a special flag that makes the resulting DVI-file contain an extra page for every `gif` environment. Furthermore, this LaTeX-run produces another file with extension *.makegif*, which contains commands to run `dvips` and `ps2gif` to to extract the interesting pages into Postscript files which are then converted to `gif` format. Obviously you need to have `dvips` and `ps2gif` installed if you want to use this feature. (A shellscript `ps2gif` is supplied with Hyperlatex. This shellscript uses `ghostscript` to convert the Postscript files to `ppm` format, and then runs `ppmtogif` to convert these into `gif`-files.)

Assuming that everything has been installed properly, using this is actually quite easy: To generate the `gif` bitmaps defined in your Hyperlatex source file *source.tex*, you run LaTeX as follows (of course you could make a shell script to save some typing).

```
latex '\def\makegifs{}\input{source.tex}'
```

This will create a DVI-file *source.dvi* and a file *source.makegif*. All `gif` images defined in *source.tex* are then created by calling

```
sh source.makegif
```

9.5 Customizing the navigation panels

Normally, Hyperlatex adds a 'navigation panel' at the beginning of every HTML node. This panel has links to the next and previous node on the same level, as well as to the parent node. The panel for the top node has a link to the first chapter or section.

In the long run, navigation panels should be fully customizable. However, since I'm still pondering how to do that properly, this isn't implemented yet. You can, however, turn the navigation panel off for selected nodes. This is done using the commands `\htmlpanel{0}` and `\htmlpanel{1}`. All nodes started while `\htmlpanel` is set to 0 are created without a navigation panel. Once the standard navigation panel has been suppressed, you can of course design and create your own navigation panel using `\link` commands.

10 Changes since Hyperlatex 1.0

Changes from 1.0 to 1.1

- The only change that introduces a real incompatibility concerns the percent sign %. It has its usual LaTeX-meaning of introducing a comment in Hyperlatex 1.1, but was not special in Hyperlatex 1.0.
- Fixed a bug that made Hyperlatex swallow certain ISO characters embedded in the text.
- Fixed HTML tags generated for labels such that they can be parsed by `lynx`.
- The commands `\+verb+` and `\=` are now shortcuts for `\verb+`*verb*`+` and `\back`.
- It is now possible to place labels that can be accessed from the outside of the document using `\xname` and `\xlabel`.
- The navigation panels can now be suppressed using `\htmlpanel`.
- If you are using LaTeX2$_\varepsilon$, the Hyperlatex input mode is now turned on at `\begin{document}`. For LaTeX2.09 it is still turned on by `\topnode`.
- The environment `gif` can now be used to turn DVI information into a bitmap that is included in the HTML-document.

Changes from 1.1 to 1.2

Hyperlatex 1.2 has a few new options that allow you to better use the extended HTML tags of the `netscape` browser.

- `\htmlrule` now has an optional argument.
- The optional argument for the `\htmlimage` command and the `gif` environment has been extended.
- The `center` environment now uses the *center* HTML tag understood by some browsers.
- The font changing commands have been changed to adhere to LaTeX2$_\varepsilon$. This hasn't been done before because it didn't make sense while font changes in HTML were not properly cumulative. The font size can be changed now as well, using the usual LaTeX commands.

Changes from 1.2 to 1.3

Hyperlatex 1.3 fixes a few bugs.

11 Acknowledgments

Thanks to everybody who reported bugs in Hyperlatex 1.0 or who suggested useful new features. This includes Arne Helme, Bob Kanefsky, Greg Franks, Jim Donnelly, Jon Brinkmann, Nick Galbreath, and Piet van Oostrum.

12 Copyright

Hyperlatex is 'free,' this means that everyone is free to use it and free to redistribute it on certain conditions. Hyperlatex is not in the public domain; it is copyrighted and there are restrictions on its distribution as follows:

13 Glossary

- **node**
  A HTML-document usually consists of several files, here called *nodes*. Other HTML documentation often calls nodes 'documents', and a full document is sometimes referred to as a 'work.'
- **preamble**
  The *preamble* of a LaTeX file is the part between the `\documentstyle` command and the `\begin{document}` command. LaTeX does not allow text in the preamble, you can only put definitions and declarations there. `hyperlatex` looks in the preamble for the commands
  - `\htmldirectory`
  - `\htmlname`
  - `\htmltitle`
  - `\htmldepth`
  - `\htmlmathitalics`
  - `\htmlautomenu`
  - `\htmladdress`
  - `\newcommand`
  - `\newenvironment`
  - `\set`
  - `\clear`

  Note that Hyperlatex will only see these commands if they start at the beginning of a line.
- **top node**
  The *top node* is the entry point of your HTML document. It is stored in a file named *basename.html*, while all other nodes are stored in numbered files (*basename_N.html*). The top node is an ancestor of all other nodes. It is considered to be at level zero, while all other nodes have a level corresponding to the sectioning command, and therefore at least 1.
  The top node often contains a menu of all sections of the document. This can be achieved using the command

  ```
  \htmlmenu{6}
  ```

References

[1] Leslie Lamport, *LaTeX: A Document Preparation System*, Addison-Wesley, 1986.

[1] at `http://www.cs.ruu.nl/people/otfried/txt/copying.txt`

HTML ⟶ LaTeX⟶ PDF, of de intrede van TeX in het hypertext tijdperk*

Yannis Haralambous

187, rue Nationale, 59800 Lille, Frankrijk
yannis.haralambous@univ-lille1.fr

Abstract

Wij beschrijven hier het produktieproces van elektronische hyper-documenten met behulp van LaTeX en *Adobe Acrobat*. Na een algemene beschrijving van de voor- en nadelen van LaTeX terzake geven we een omschrijving van elke stap, alsook van zekere te nemen voorzorgen met als doel efficiënte *Acrobat* documenten te bekomen.
De lezer zal in dit artikel een beschrijving vinden van de software tools DVIPS `repere` en `recticrt`, en eveneens de basisprincipes van het formaat PDF.

Sleutelwoorden: Acrobat, DVIHPS, PDF, `repere`, `recticrt`

1 Inleiding

1.1 Hypertext en LaTeX wat is het verband?

In tegenstelling met *hyper*markt, *hyper*tentie, *hyper*activiteit, *hyper*trofie en de *hyper*snelheid, waar het voorvoegsel '*hyper*' de hoge, zelfs overdadige hoeveelheid aanduidt, is *hypertext* geen zeer uitgebreide tekst, maar een tekst die gekenmerkt wordt door een interne structuur die aan bepaalde schermbesturingsprogramma's toelaat om door de pagina's van het document te navigeren.

Er is dus een triviaal verband tussen het begrip hypertext en de markeringen in een LaTeX document: beide voegen structuur toe aan het document. Bijvoorbeeld, het begrip van indexeren binnen LaTeX stemt volledig overeen met het begrip van hypertext-verband.

Het essentieel verschil tussen beide concepten is de desinteresse van TeX ten opzichte van de schermuitvoer. Inderdaad, TeX manipuleert hokjes welke bestemd zijn om karakters of tekeningen te bevatten. De taak van het invullen van deze hokjes met karakters/tekeningen valt ten deel aan de verschillende drivers voor scherm, fax of printer. TeX is in de eerste plaats een werktuig voor typografische opmaak, het scherm dient enkel om te 'previewen', en een afbeelding op scherm wordt nooit als einddoel van een TeX-compilatie beschouwd.

Deze desinteresse van TeX t.o.v. het scherm is des te belangrijker vermits de PostScript constructies, welke in een DVI file ingevoerd zijn met behulp van het SPECIAL commando, in het algemeen niet zichtbaar zijn op het scherm.[1] De schermafbeelding is in die gevallen des te meer ontgoochelend vergeleken met het afgedrukte resultaat.

Er blijkt dat, voor de eerste maal in zijn bestaan, TeX uitermate nuttig is voor het aanmaken van documenten die bedoeld zijn om vanaf het scherm te worden gelezen. LaTeX is inderdaad volledig geschikt voor de automatische produktie van hypertext verbanden, en de methodes waarover wij zullen spreken binnen dit artikel laten een automatische conversie naar een hypertext document toe van elk reeds bestaand LaTeX document! Het is passend te benadrukken dat een dergelijk document zijn volledige typografische LaTeX kwaliteit bewaart en afgedrukt kan worden op dezelfde wijze als voordien.

1.2 Het algemeen schema

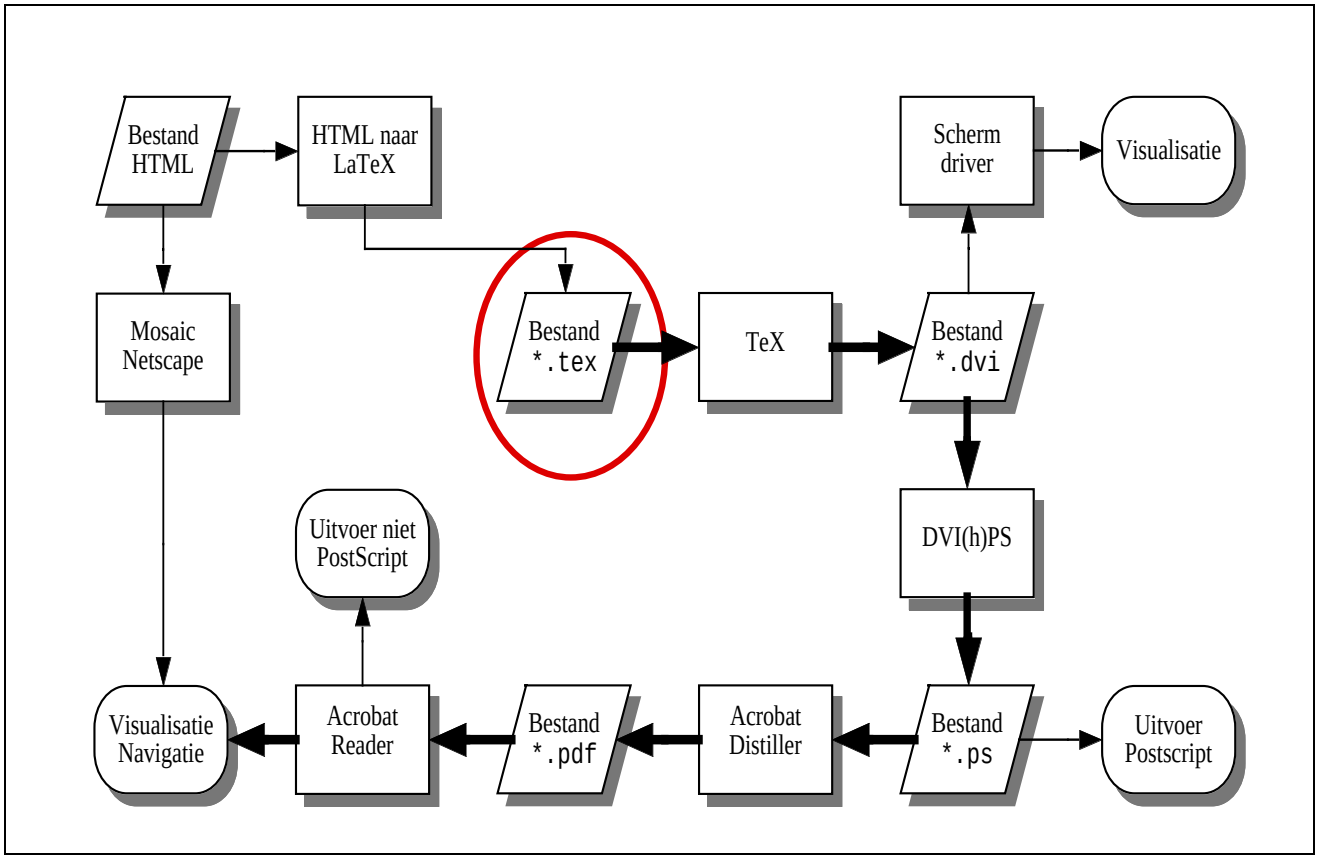
TeX en LaTeX lezen een bestand in dat voorzien is van structurele en visuele markeringen en creëren een tweede bestand dat de te drukken pagina met een grote nauwkeurigheid omschrijft. Dit uitvoerbestand van TeX wordt DVI genoemd (*DeVice Independent* = platform-onafhankelijk) omdat het enkel abstracte gegevens bevat: de plaatsing van elk karakter op de pagina, de naam van het font waarin de driver de pixels zal vinden van dit karakter, zijn code binnen dit font, enzovoort.

---

*Dit artikel stemt overeen met de presentatie gegeven te Nanterre (F) op 19 januari 1995 tijdens de dag *Verspreiding van electronische documenten*.

Eerder gepubliceerd in **Cahiers GUTenberg**, #19, Januari 1995.
Vertaling van het Frans naar het Nederlands: **Philippe Vanoverbeke**, Hameau de Penfrat, F-29160 Crozon, Frankrijk; Langenhoekstraat 28, B-8210 Veldegem, België. Met veel dank van de MAPS redactie aan Philippe voor het moeilijke vertaalwerk, vermits 'Ik heb in sommige termen gewoonweg in het Engels vertaald vermits dit zo gebruikelijk is (dacht ik). In Frankrijk heeft men echter wel voor alles (godbeterdhet) een Franse uitdrukking. . . ', aldus Philippe.

[1] Met uitzondering van de gelukkigen onder ons die een operating system gebruiken voorzien van *Display PostScript*.



Figuur 1: *Het algemene schema*

Het visualiseren/afdrukken van een DVI bestand veronderstelt dus de beschikbaarheid over een aantal fonts. Dit is altijd mogelijk in het kader van informaticasystemen verbonden via een netwerk (werkstations, mainframes), doch dit wordt problematisch in het kader van de personal computers. De situatie wordt nog erger wanneer men documenten elektronisch wenst te verspreiden: een document bedoeld om gevisualiseerd en afgedrukt te worden door een groot aantal personen kan moeilijk in DVI formaat worden verspreid (dit zou enkel de TeX gemeenschap betreffen, en bovendien zou men zich dienen te beperken tot het voorbijgestreefde CM font dat nog altijd zeer verspreid is in de V.S.). Meer bepaald zou het onmogelijk zijn een document op een floppy te plaatsen met voldoende utilities opdat het document onmiddellijk zou kunnen worden gevisualiseerd of afgedrukt worden zonder dat een TeX systeem is geïnstalleerd.

Tenslotte zijn de hypertext verbanden niet voorzien binnen de syntax van het DVI formaat; elke poging tot ontwikkeling van een hypertext driver die het DVI formaat gebruikt zal er ons toe leiden een nieuw formaat 'Hyper-DVI' te definiëren, met alle problemen van compatibiliteit van dien en de terughoudendheid van de TeX gemeenschap die fier is op de stabiliteit van zijn werkinstrumenten. Er blijkt dus dat het DVI formaat niet de ideale kandidaat is voor een bestandsformaat dat snel uit te baten en voldoende interactief is om de integratie van hypertext verbanden toe te laten.

Wat te doen dan? De voor de hand liggende keuze van een dergelijke kandidaat — vandaag tenminste — is het PDF formaat (*Portable Document Format*) van Adobe Systems. Er bestaat een uitbreiding van de PostScript taal, zeer nauw aansluitend met de syntax van bestanden aangemaakt met het programma Illustrator, met twee grote vernieuwingen t.o.v. de pure PostScript taal: het visualiseren en afdrukken op eender welk platform, onafhankelijk van de gebruikte fonts binnen het document en de integratie van de functionaliteit van hypertext.

We zullen het nog later hebben over het formaat PDF (zie 6.1). Laten we bekijken hoe het gebruik van dit formaat past binnen het proces van de aanmaak van een document (elektronisch of gedrukt). In het schema van figuur 1 zijn de in-uit bestanden aangeduid met schuine symbolen, de programma's met rechthoeken en de bewerkingen (visualisatie, afdrukken enz.) door symbolen met afgeronde hoeken. De vet afgedrukte pijlen geven de te volgen weg aan waaraan dit artikel is gewijd.

Het `*.tex` bestand is ons vertrekpunt (aangeduid met een cirkel). Vertrekkend van dit bestand (alsook van de style bestanden en de fontgegevens) zal TeX een `*.dvi` bestand produceren. Indien men gebruik heeft gemaakt van het LaTeX formaat en men de extensie `hyperref` erin opgenomen heeft, zullen de commando's voor cross-references, bibliografische aanduidingen, samenvatting en indexen de hypertext verbanden produceren binnen het `*.dvi` bestand met behulp van `\special` invoegingen.

Een andere te beschouwen mogelijkheid als vertrekpunt is een HTML bestand (bovenaan links op ons schema). Een HTML bestand kan inderdaad zeer gemakkelijk worden geconverteerd in een LaTeX bestand en de hypertext verbanden binnen deze eerste kunnen bewaard blijven binnen de tweede.

Laten we teruggaan naar ons `*.dvi` bestand. Dit kan meteen uitgebaat worden d.m.v. een scherm driver of een niet-PostScript printer. Maar men kan het tevens converteren naar PostScript m.b.v. het programma DVIPS. Een uitgebreide versie van DVIPS, met name DVIHPS ('DVI naar *hyper*PostScript') bewaart de eventuele hypertext verbanden welke het `*.dvi` bevat. Eenmaal het PostScript bestand aangemaakt kan men dit afdrukken op een PostScript printer (of op een niet-PostScript printer d.m.v. Ghostscript) of converteren naar PDF formaat m.b.v. het programma Adobe Acrobat Distiller. Deze laatste zal de hypertext verbanden interpreteren en ze opnemen in het PDF document.

Om tenslotte het PDF document te visualiseren, gebruikt men Adobe Acrobat Reader, een programma dat gratis wordt verspreid en geschikt is voor Macintosh, Windows, DOS en Solaris. Dit programma laat ons tevens toe om te navigeren in documenten en deze af te drukken op een niet-PostScript printer.

Men merkt dat als het vertrekpunt een HTML document is, de hypertext functionaliteit bewaard is gebleven, maar dat men er bovenop de typografische presentatie van LaTeX er bij heeft gekregen. Een PDF document is een waarheidsgetrouwe copie van het gedrukte document (het kan op film gezet worden en zo een professionele afdruk leveren met kleurenfoto's, grafieken enz.) die als extra de hypertext navigatiemogelijkheid biedt in het document of over het net (de navigatie over het net is enkel mogelijk met versie 2 van de Adobe Acrobat software).

1.3 Conclusies

De documentstructuur vereist door LaTeX wordt binnen steeds meer domeinen toegepast: het meest frappante voorbeeld is ongetwijfeld deze van de vocale synthesiser (zie [5]), voor gebruik door niet-zienden, die in staat is een wiskundige formule uit te spreken zoals een wiskundige en toelaat doorheen het document te navigeren met behulp van speciale toetsen op de recorder.

In dit artikel zullen we een andere applicatie omschrijven: de creatie van electronische boeken waarvan de presentatie niet hoeft onder te doen voor traditionele boeken (vermits ze kunnen worden afgedrukt zonder verlies van kwaliteit) en die een minimum aan interactiviteit bieden: hypertext navigatie tussen samenvatting, index, bibliografische referentielijst en tekst binnen de tekst zelf met behulp van LaTeX cross-references, en tenslotte navigatie tussen documenten aanwezig op eenzelfde machine of op het netwerk.

In de rest van dit artikel volgen alle stappen van het proces, aangeduid door vetgedrukte pijlen op het schema van fig. 1 (en op deze van fig. 2 die er een uitbreiding van vormt).

2 HTML ⟶ LaTeX

Het HTML mark-up systeem is gedefinieerd volgens de SGML norm (de lezer zal de volledige DTD vinden in [3]). Het betreft een klein aantal markeringen, hoofdzakelijk bedoeld voor de weergave op een scherm. Hierdoor vind men diverse tekst stijlen, zowel logische (emphasized, very emphasized, citation, address enz.) als visuele (schuin, vet, onderlijnd enz.), maar alsook markeringen voor fundamentele strukturen als voetnoten, tabellen, enz.

Het spreekt voor zich dat LaTeX een oneindig rijker mark-up systeem is dan HTML en bijgevolg is de conversie 'HTML ⟶ LaTeX' triviaal. Inderdaad, het betreft eenvoudigweg de conversie van:

1. bepaalde markeringen binnen LaTeX zoals `<CITE>` in `\begin{quotation}` en `</CITE>` en `\end{quotation}`
2. andere markeringen in LaTeX commando's met een argument: `<EM>` een woord `</EM>` in `\emph{een woord}`, enz.
3. enkele zeldzame markeringen in LaTeX commando's zonder argument zoals `<P>` die geconverteerd kan worden in een lege regel of in `\par`
4. geaccentueerde letters in LaTeX syntax: `&eacute;` zal `\'{e}` geven, `&Ccedilla;` geeft `\c{C}` enzovoort.

Twee particuliere gevallen doen zich voor:

1. De HTML markeringen die geen onmiddellijk LaTeX equivalent hebben, bijvoorbeeld `<STRONG>` die een sterk emphasized omgeving aanduidt. In dit geval zou de definitie van nieuwe LaTeX omgevingen de aan te raden oplossing zijn, eventueel met behoud van de dezelfde benaming en de keuze van de visuele presentatie ervan aan de gebruiker over te laten, dit met een default presentatie voorhanden.
   Zo zou bijvoorbeeld
   `<STRONG>very emphasized</STRONG>`
   kunnen geconverteerd worden in:
   `\begin{STRONG}sterk benadrukt\end{STRONG}`
   deze omgeving zou kunnen gedefinieerd worden als
   ```
   \newenvironment{STRONG}{%
                       \bfseries\itshape}{\/}
   ```
   in een bestand `html2ltx.sty` en die dus '**sterk benadrukt**' oplevert in een LaTeX document.
2. De markeringen van hypertext functies. In dit geval volstaat het om de syntax, aangewend door de extensie `hyperref`, die we vervolgens (zie 3) zullen gebruiken om precies deze informatie in het DVI bestand op te nemen. Hier volgen de expliciete commando's:
   1. om een *doel* te definiëren (een 'anker' in het HTML jargon), zal men onder HTML de gekozen tekst omgeven door de markeringen `<A NAME="sleutelwoord">`(waar `sleutelwoord` het sleutelwoord is gegeven aan het doel in kwestie), en `<\A>`. Het overeenkomstig LaTeX commando zal `\hyperdef{}{sleutelwoord}{}{...}` zijn waar . . . de gekozen tekst is;

2. om een *verband* met een doel te leggen binnen hetzelfde document zal men in HTML de gekozen tekst omgeven met de markeringen `<A HREF="#sleutelwoord">` (waar `sleutelwoord` het sleutelwoord is gegeven aan het doel in kwestie), en `<\A>`. Het overeenkomstig LaTeX commando zal `\hyperref{}{sleutelwoord}{}{...}` zijn waar ... de gekozen tekst is;
3. om een *verband* te leggen naar een ander document zal men in HTML de gekozen tekst omgeven met de markeringen `<A HREF="adres">` (waar `adres` het `att`padres is van het document in kwestie), en `<\A>`. Het overeenkomstig LaTeX commando zal `\hyperref{adres}{}{}{...}` zijn waar ... de gekozen tekst is.

3 LaTeX −→ DVI

Laten we het meteen vertellen: elk LaTeX2_ε, zonder uitzondering, kan een elektronisch document produceren door de toevoeging van één regel:

```
\usepackage{hyperref}
```

vooraan in het bestand. Deze uitbreiding werd ontwikkeld door Tanmoy Bhattacharya en gewijzigd door Sebastian Rahtz. De LaTeX commando's die zullen dienen om hypertext verbanden te creëren zijn:

- `\label`, `\ref` en `pageref` (cross-references);
- `\chapter`, `\section`, `\subsection`, etc. (indeling van het document);
- `\index` (aanmaak van een index);
- `\cite` (aanmaak van een lijst met bibliografische referenties).

Er dient dus niets gewijzigd te worden in de brontekst van het LaTeX document, tenzij men er nieuwe hypertext functionaliteiten wenst aan toe te voegen waarbij in dit geval de interne commando's `hypertarget` en `\hyperlink` kunnen gebruikt worden, of door nieuwe commando's te definiëren gebruik makend van de 'private' commando's `hyper@anchor` en `hyper@link` (zie [4] voor meer details).

3.1 De aanmaak van een PDF bestand met merkteken

Zoals men kan zien op fig. 3 geeft het programma Acrobat Reader ons de mogelijkheid een inhoudstafel te hebben die hiërarchisch is en interactief op het linkergedeelte van het Acrobat venster. Het programma DVIHPS — tenminste in zijn huidige versie — maakt deze inhoudstafel niet automatisch aan. Om dit tekort te verzachten heeft de auteur een 'utility' ontwikkeld, `repere` genaamd, samen met een LaTeX style (`repere.sty`). Een een 'executable' van `repere` op het eigen platform te bekomen dient de gebruiker eerst de 'utility' Flex (versie $\geq$ 2.4.6) los te laten op `repere.lex`, met als optie `-8` in de commandoregel:

```
flex -8 repere.lex
```

en vervolgens een C compilatie van het resultaatbestand van Flex (`lex.yy.c` onder UNIX of Macintosch, `lexyy.c` onder DOS enzovoort).

In fig. 2 zal de lezer het algemeen schema van fig. 1 vinden waaraan de tussenstap `repere` is toegevoegd.

Ziehier de exacte procedure om een PDF inhoudstafel aan te maken:

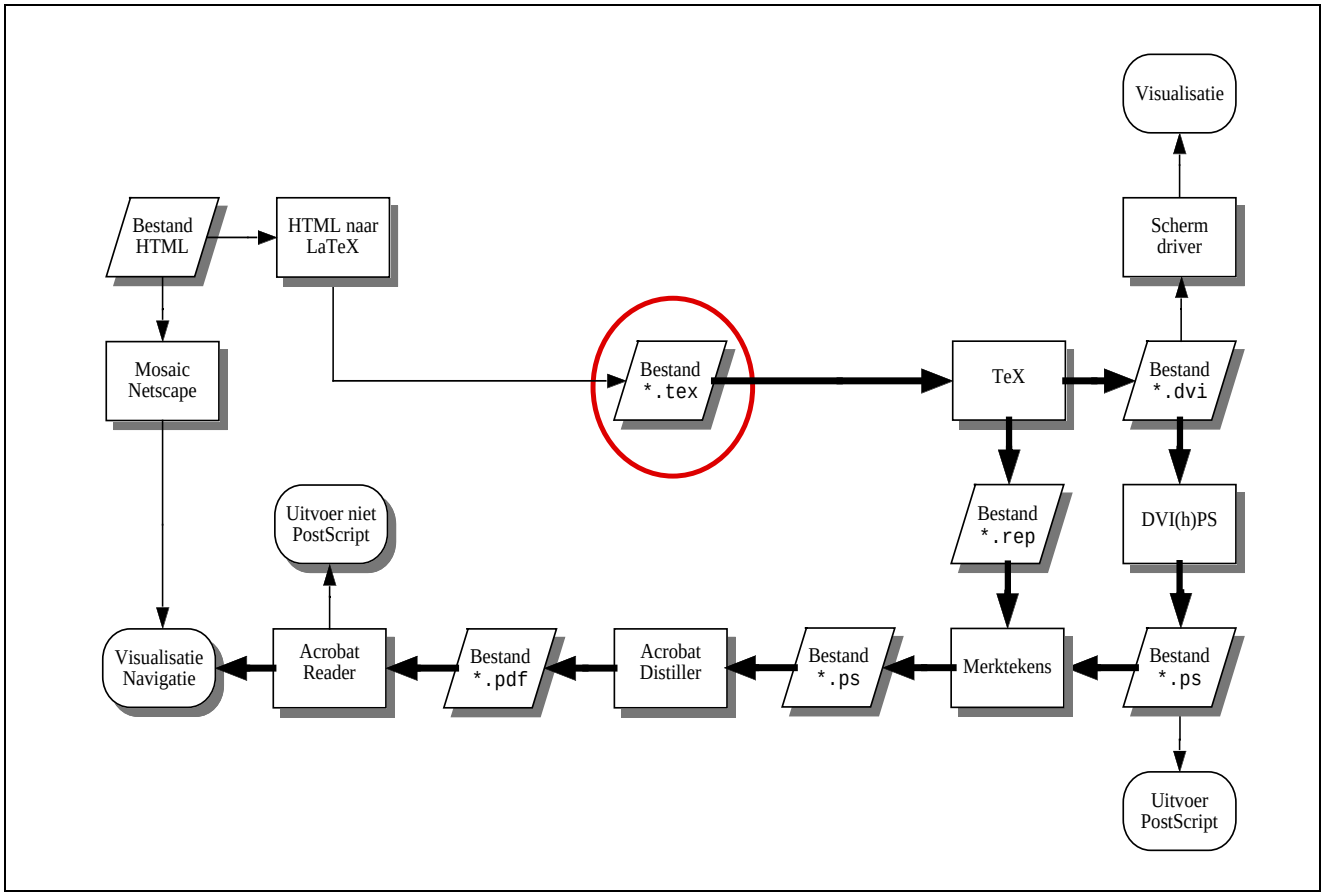
- men voegt het commando `\usepackage{repere}` toe in het begin (preamble) van het document, *na* het commando `\usepackage{hyperref}`;
- men vervangt de commando's `\part`, `\section`, `\subsection` en `\subsubsection` die men in de inhoudstafel wil opnemen door respectievelijk `\Part`, `\Section`, `\Subsection` en `\Subsubsection`;
- LaTeX zal dan een nieuw auxiliair bestand aanmaken met extensie `.rep`;
- Na DVIHPS start men `repere` op met als invoerbestand vervolgens:
  1. het bestand `foo.rep` aangemaakt door LaTeX;
  2. het bestand `foo.ps` aangemaakt door DVIHPS;
  3. en opnieuw het bestand `foo.rep`.

  Onder UNIX (en onder elk ander operating system dat over 'pipes' beschikt) kan dit heel eenvoudig gebeuren met behulp van een commandoregel van het type
  ```
  cat foo.rep foo.ps foo.rep | repere >
                                  new-foo.ps
  ```
- Het bestand `new-foo.ps` aangemaakt door `repere` is een exacte kopie van het bestand `foo.ps` met enkele extra regels welke Acrobat Distiller toelaten om een nieuwe PDF inhoudstafel te creëren.

Het meest delicate gedeelte van deze operatie is de conversie van de verschillende tekstonderverdelingen naar de PDF codering. Inderdaad, deze codering is een samenraapsel van Macintosh Standard, Adobe Standard en Windows ANSI coderingen. De moeilijkheid vloeit voort uit het feit dat LaTeX een uitbreiding van de commando's maakt wanneer hij reeksen karakters wegschrijft in het bestand `*.rep\verb`. Een geaccentueerd karakter kan dus op verschillende wijzen omschreven worden, afhankelijk van de codering komende van LaTeX de accent-commando's enz. De 'utility' `repere` kan moeilijk elke schrijfwijze herkennen en deze vervangen door het overeenstemmende 8-bits PDF karakter. Het kan dus voorvallen dat de gebruiker enkele correcties dient uit te voeren in de titels van de inhoudstafel van het PDF bestand. Het is dus mogelijk dat de gebruiker enkele correcties dient uit te voeren in de titels van de onderverdelingen van de PDF inhoudstafel. Gelukkig heeft dit geen enkele invloed op de interactiviteit: de hypertext relaties tussen deze inhoudstafel en de tekst van het PDF document zullen bewaard blijven.

3.2 Voorzorgsmaatregelen te nemen op het TeX niveau

Het feit dat een DVI bestand totnogtoe bijna uitsluitend diende voor het afdrukken, verplicht ons bepaalde voorzorgen te nemen bij de voorbereiding van dit bestand wanneer



Figuur 2: *Algemeen schema om een bestand PDF met merktekens te verkrijgen*

het onze bedoeling is dit later te converteren naar het PDF formaat.

Deze voorzorgen betreffen hoofdzakelijk de keuze van het in het document gebruikte lettertype. Inderdaad, het grootste probleem voor een programma zoals Acrobat, dat beweert eender welk PostScript bestand te kunnen vertonen en afdrukken, is de beschikbaarheid van de PostScript lettertypes die in het document gebruikt worden. 99% van de bestaande PostScript lettertypes (er zijn er duizenden. . .) zijn commercieel en het gebruik ervan houdt dus een voorafgaandelijke transactie in tussen de gebruiker en het bedrijf dat de rechten ervan bezit. Wat te doen dan om een dergelijk document te verdelen? Hoe kan men zeker zijn dat elke lezer de beschikking heeft over dezelfde PostScript lettertypes? Wat te doen indien dit niet het geval is?

Adobe heeft dit probleem opgelost door een nieuwe techniek voor lettertypes te ontwikkelen: de *Multiple Master* lettertypes. Het principe is aardig verwant met METAFONT: het betreft meta-lettertypes[2] en door de karakters enigszins te wijzigen is de scherm- en printerdriver (SuperATM) in staat niet-beschikbare PostScript lettertypes na te bootsen. Twee Multiple Master lettertypes vergezellen Acrobat Reader: een ‘roman’ lettertype en een ‘sans serif’. Elk onbeschikbaar lettertype zal vervangen worden door een van deze twee ‘generieke’ lettertypes.

Elk onbeschikbaar lettertype? Ja, elk onbeschikbaar lettertype, inclusief alle wiskundige symbolen of speciale karakters (fonetisch, niet-latijnse, enzovoort). Door gebruik te maken van niet-standaard lettertypes kan men dus snel tot rampzalige resultaten komen.

Om hieraan te verhelpen stelt Acrobat Distiller twee oplossingen voor:

1. de opname van het ‘exotisch’ PostScript lettertype binnen het PDF bestand. Dit voorkomt de substitutie door Multiple Master maar stelt eventueel problemen op het vlak van het copyright;
2. het gebruik van (niet-PostScript) PK fonts in het geval van TeX. Er is geen copyright probleem vermits enkel enige bitmaps opgenomen zullen worden in het PDF bestand maar de schermafdruk onder Acrobat Treader lijdt hieronder. Inderdaad, Acrobat Reader is niet geoptimaliseerd om bitmap karakters te vertonen en moet vaak overgaan op een grote vergroting om het ontcijferen toe te laten. Daarentegen gebeurt het afdrukken zonder probleem (vermits uiteindelijk de resolutie van de bitmap karakters opgenomen in het bestand eerder overeenkomen met de resolutie van de printer dan deze van de monitor).

Een derde oplossing bestaat uit het vermijden van het probleem door enkel ‘universele’ lettertypes te gebruiken, t.t.z.

[2] In feite zijn hun meta-eigenschappen lachwekkend vergeleken bij deze van de METAFONT lettertypes.

ongewijzigd aanwezig onder elk operating system: zodoende is men niet aangewezen op de opname van lettertypes in het document (en bijgevolg stellen er zich geen copyright problemen) maar men behoudt de PostScript vertoning op het scherm die gevoelig beter is dan de vertoning van bitmap karakters. Deze lettertypes zijn Times, Helvetica, Courier, Symbol, Palatino, New Century Schoolbook en Bookman.

Uiteindelijk een ultieme voorzorg: men dient het gebruik van *virtuele fonts welke gebruik maken van samenvoegingen van tekens* te vermijden. Anders gezegd: er dient gebruik gemaakt worden van her-coderingen van lettertypes 'à la DVIPS' eerder dan gebruik te maken van de samenstelling van geaccentueerde karakters door de combinatie van de letter en het accent zoals `fontinst.sty` dit doet. De reden hiervoor is dat Acrobat toelaat om karakter-strings op te zoeken in een document: indien de geaccentueerde karakters in werkelijkheid samenvoegingen zijn van niet-geaccentueerde letters en accenten zal men geen zoekslag kunnen uitvoeren op basis van 8-bits geaccentueerde karakters. Het woord 'dénégré' bijvoorbeeld zal gecodeerd worden als

```
de<accent aigu>ge<accent aigu>ne<accent
                          aigu>re<accent aigu>
```

in het PDF document en het opzoeken van de karakterstring `d\'eg\'en\'er\'e` (waar '`\'e`' een 8-bits Macintosch of Windows of ISO Latin-1 karakter is) is tot mislukken gedoemd.

Om virtuele fonts te bekomen *die zich beperken tot hercodering van karakters*, kan men gebruik maken van de utility `afm2tfm` van de DVIPS distributie, met de commandoregelopties `-T extex.enc` (Cork encoding) en `-v` gevolgd door de naam van het gewenste VPL bestand. In het configuratiebestand `psfonts.map` van DVIPS moeten bijgevolg de volgende commando's bijgevoegd worden:

```
" ExtendedTeXEncoding ReEncodeFont " < extex.enc
```

aan de font definities waarvan sprake (veronderstellend dat het bestand `extex.enc` zich in dezelfde directory bevindt als `psfonts.map`, anders dient `extex.enc` vervangen te worden door zijn volledig pad).

Ziehier een voorbeeld: de auteur heeft het reëel font `rtimes.tfm` en het virtueel font `vtimes.vf` (met metriek bestand `vtimes.tfm`) verkregen vanuit het bestand AFM `Times-Roman.AFM`, met behulp van de volgende commando's:

```
afm2tfm Times-Roman.afm -T extec.enc
                        -v vtimes.vpl rtimes.tfm
vftovp vtimes.vpl -o vtimes.vf
```

Om het te gebruiken heeft hij de regel

```
rtimes Times-Roman " ExtendedTeXEncoding
                     ReEncodeFont " <extec.enc
```

toegevoegd aan het bestand `psfonts.map`, en heeft hij een bestand `T1vtime.fd` gecreëerd dat, onder andere, de volgende regels bevat:

```
\DeclareFontFamily{T1}{vtime}{}
\DeclareFontShape{T1}{vtime}{m}{n}{<->vtimes}{}
\endinput
```

Om deze nieuwe font familie te gebruiken volstond het de volgende regels op te nemen:

```
\usepackage{t1enc}
\def\rmdefault{vtime}
```

in het begin van het LaTeX bestand.

Voor de wiskundige formules kan men de fonts Times en Symbol gebruiken: meerdere makro- en virtuele fontpakketten laten dit toe, onder andere *MathTimes* van Michael Spivak.

4 DVI −→ (Hyper)PostScript

Zoals TeX is DVIPS een typisch voorbeeld van een public domain programma van zeer hoge kwaliteit. Unaniem erkend als de beste vertaler van DVI naar PostScript zal DVIPS hoogstwaarschijnlijk de norm worden op dit vlak. Overigens bieden steeds meer TeX systemen een schermdriver aan (of een niet-PostScript printer) specifiek voor het onderliggende operating system en een implementatie van DVIPS voor PostScript afdruk.

Mark Doyle heeft het programma DVIPS (van Tom Rokicki) gewijzigd (aangespoord door *[sic]* Paul Ginsparg en Arthur Smuth) opdat het het beheren van hypertext verbanden zou toestaan. Was een dergelijke 'chirurgische' ingreep noodzakelijk? Het blijkt van wel en we zullen zien waarom.

Om een hypertext verband te definiëren heeft Acrobat Distiller (op zijn minst) twee gegevens nodig: de actieve zone bij het begin en het scherm bij aankomst van het verband. Deze informatie dient hem medegedeeld te worden onder de vorm van PostScript coördinaten (1/72 inches vanuit de oorsprong van de PostScript coördinaten, onderaan links van de bladzijde).

Om de exacte coördinaten van het doel van een hypertext verband te determineren, welke zich meerdere pagina's na de actieve beginzone kunnen bevinden *moet* men post-processing toepassen op het DVI bestand: het zou zoniet onmogelijk dan wel op zijn minst enorm ingewikkeld zijn deze coördinaten op het niveau van LaTeX te bekomen. Vermits een DVI post-processor benodigd is kan men evengoed meteen DVIPS gebruiken, deze werkt met PostScript coördinaten en houdt rekening met alle schaalveranderingen en translaties van de oorsprong welke kunnen voorkomen.

Zodoende heeft Mark Doyle een 'opgeknapte' versie van DVIPS ontwikkeld welke hij DVIHPS noemt ('DVI naar *Hyper*PostScript'). De auteur hoopt dat deze wijzigingen binnenkort deel zullen uitmaken van de DVIPS standaard.

Praktisch gezien wordt DVIHPS op dezelfde wijze gebruikt als DVIPS op een uitzondering na: indien de gebruiker een PostScript bestand wenst aan te maken waarin de hypertext informatie ten behoeve van Acrobat Distiller is opgenomen, dan zal hij de optie `z` (kleine z) dienen toe te voegen aan de commandoregel.

Er dient opgemerkt dat de header van het aangemaakte PostScript bestand PostScript code zal bevatten die alle

hypertext commando's desactiveert indien het bestand geïnterpreteerd wordt door een andere PostScript interpreter dan Acrobat Distiller. Diezelfde code zal tevens de versie van Acrobat Distiller nagaan (bijvoorbeeld: versie 1 van dit programma staat geen verbanden toe tussen verschillende documenten: enkel deze verbanden worden gedesactiveerd gedurende de interpretatie).

5 PostScript −→ PDF

Deze etappe, welke zeker de langste is op het vlak van de benodigde tijd voor de uitvoering ervan, wordt volledig afgehandeld door het programma Acrobat Distiller (een eerder duur programma, doch ongelukkiglijk onmisbaar om degelijke PDF documenten te produceren). Acrobat Distiller is tegelijkertijd een zeer goede 'debugger' voor PostScript code en een goede interpreter: zonder een PostScript kleurenprinter tot zijn beschikking te hebben, noch het Display PostScript systeem om PostScript te kunnen previewen op een scherm, gebruikt de auteur Acrobat Distiller om TeX documenten in kleur te visualiseren.

6 Previewen, navigatie en afdruk van het PDF bestand

Deze operaties gebeuren met behulp van het programma Adobe Acrobat Reader, hetgeen gratis wordt verdeeld.[3] Meerdere functies voor zoeken, navigatie, kopiëren van text[4], enz. worden de gebruiker aangeboden onder de vorm van 'buttons' en menus. Er dient opgemerkt dat dit programma werkt op Macintosh, alsook onder DOS, Windows en UNIX (Solaris en binnenkort andere systemen), met exact dezelfde gebruikers interface.

Om de scherm-afdruk van een TeX bestand te vergelijken kan de gebruiker in figuur 3 een kopie van een scherm van Acrobat Reader op Macintosh vinden en in figuur 4 de afgedrukte versie van hetzelfde document. De LaTeX uitbreiding `hyperref` laat de gebruiker toe de presentatie van de actieve zone van de hypertext link te kiezen (by default in het rood) alsook de doel-zones (by default groen). Het PDF formaat laat onder andere toe de actieve zones te omlijnen.

6.1 Enkele inlichtingen betreffende het PDF formaat

Voor de gewone sterveling is het PDF formaat nog ontoegankelijker en onleesbaarder dan de PostScript taal zelf. Niettemin is het interessant om enige kennis te hebben over de structuur ervan om, indien nodig, enige kleine aanpassingen te kunnen uitvoeren aan een dergelijk bestand (het PDF formaat is zeer recent van oorsprong en er is een zeer groot gebrek aan utilities om PDF documenten te kunnen wijzigen).

Een PDF bestand is een tekstbestand, geheel geschreven in 7-bit. Het bestaat uit vier delen: de *header*, de *body*, de *cross-reference table* en de *trailer*. In de versie 1 bevat de header één enkele regel: `%PDF-1.0`. De body is samengesteld uit objecten: elke pagina is een object, de verbanden, de nota's, de markeringen, de fontcodering, de fontomschrijvingen alsook de omschrijvingssystemen t.b.v. kleur zijn objecten. Het voordeel om objecten te gebruiken vloeit voort uit het feit dat men hierdoor pagina's kan bijvoegen, verwijderen, de volgorde ervan kan wijzigen, zonder de bestaande hypertext verbanden te verbreken: de paginavolgorde is bewaard in de cross-references tabel, verwijderde pagina's blijven dus in het document behouden en worden enkel 'virtueel' verwijderd. Elke wijziging brengt met zich de aanmaak van een nieuwe tabel met cross-references mee in dit in de trailer van het document. De applicaties voor de visualisatie van PDF documenten beginnen dus een dergelijk document vanaf het einde in te lezen en halen in de cross-references tabel de pointers op naar de diverse objecten van het document.

De meeste objecten zijn gecomprimeerd en daarna gecodeerd in 7-bit code: vier comprimatiemethodes kunnen aangewend worden: Lempel-Ziv, *run length*, CCITT Fax groep 3 of 4, JPEG; voor de codering naar 7-bit code kunnen twee methodes gebruikt worden: de hexadecimale notatie of de 'modulo 85' notatie.

Met Acrobat Distiller kan dit comprimeren gedesactiveerd worden, doch dit biedt weinig voordelen vermits geen enkele utility voorhanden is die het comprimeren *a posteriori* van gewijzigde PDF documenten toelaat.

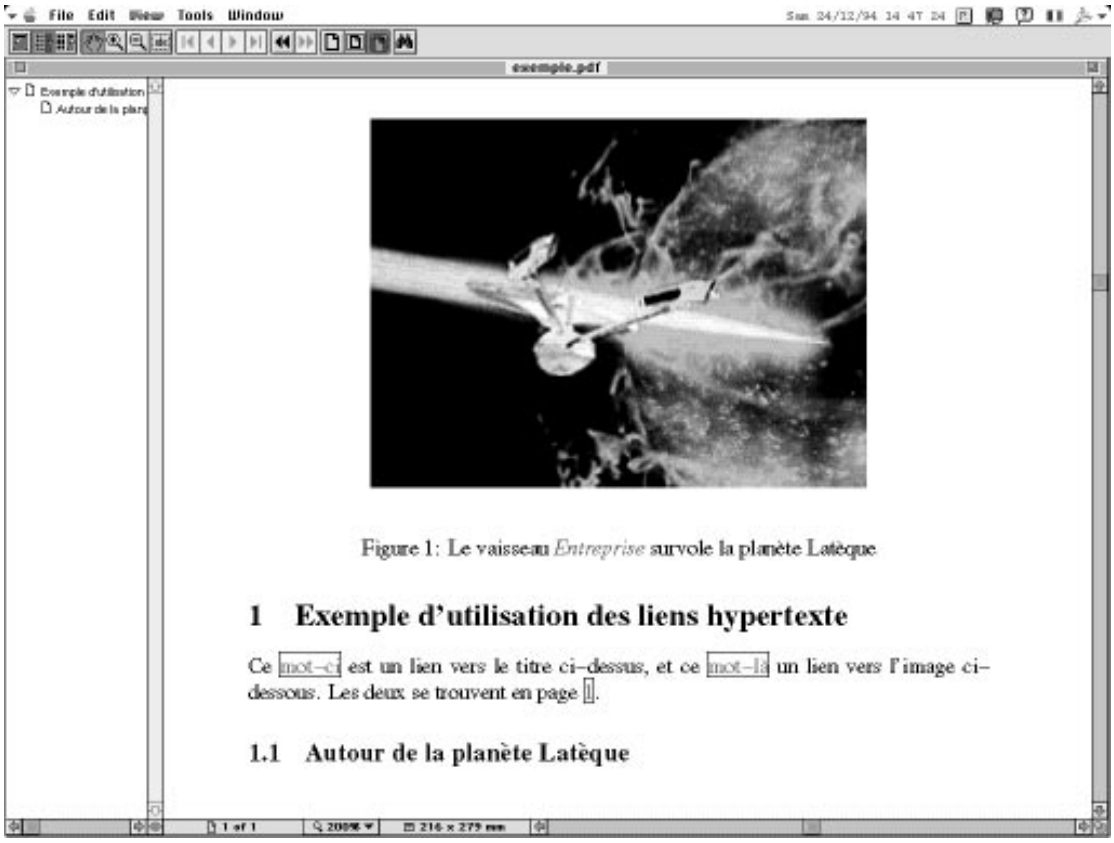
We beschrijven hier verder uitsluitend enkele niet-gecomprimeerde objecten welke bijgevolg door de gebruiker vrijelijk kunnen gewijzigd worden. Niettemin dient te worden opgemerkt dat elke wijziging van het PDF bestand (met één uitzondering, die we verder behandelen) de verplichting met zich meebrengt om de cross-references tabel aan te passen: inderdaad, deze tabel bevat voor elk PDF object zijn verschuiving in positie van *bytes* ten opzichte van het begin van het bestand. Elk object heeft een nummer dat tevens het eerste gegeven is van een object. De objecten zijn in het PDF bestand niet noodzakelijk geordend in stijgende lijn van dit nummer.

De tabel met cross-references bevat een regel voor elk object; deze regel bevat de verschuiving van het object t.o.v. het begin van het bestand (getal met 10 cijfers) gevolgd door een lege spatie en een getal met 5 cijfers dat aanduidt hoe vaak het betreffende object gewijzigd is geworden, opnieuw gevolgd door een lege spatie en dan de letter 'n'. Indien het object gewist word is het objectnummer dus beschikbaar op deze regel doch de syntax wordt gewijzigd: het getal met 10 cijfers duidt het nummer aan van het volgende vrije object in de tabel (=0 indien het hier het laatste vrije object betreft) en de letter op het einde van de regel is 'f'.

Voor elke wijziging uitgevoerd aan een object dient de resulterende verschuiving doorgegeven te worden aan de opeenvolgende objecten in het bestand. Tevens dient een

[3] En op de Adobe demo CD-ROM welke bij deze MAPS is bijgesloten.

[4] En het zal de eerste maal zijn dat men kopieën van tekst kan maken vanuit een afgeleide van een DVI bestand.



Figuur 3: *Schermkopie Macintosh systeem: een voorbeeld van een PDF bestand aangemaakt onder LaTeX*

Figure 1: Le vaisseau *Entreprise* survole la planète Latèque

1 Exemple d'utilisation des liens hypertexte

Ce mot-ci est un lien vers le titre ci-dessus, et ce mot-là un lien vers l'image ci-dessous. Les deux se trouvent en page 1.

1.1 Autour de la planète Latèque

Figuur 4: *Resultaat van de afbeelding van het PDF bestand*

getal op het einde van het bestand gewijzigd te worden, dit getal duidt de verschuiving aan van de cross-references tabel ten opzichte van het begin van het bestand.

Een voorbeeld: het bestand uit fig. 3 bevat 20 objecten; ziehier de twee laatste, de cross-reference table en de trailer:

```
1 0 obj
<<
/CreationDate (Dim 25 D\'ec 1994)
/Producer (Acrobat Distiller 1.0 pour Macintosh)
>>
endobj
3 0 obj
<<
/Pages 5 0 R
/Outlines 16 0 R
/Type /Catalog
>>
endobj
xref
0 21
0000000000 65535 f
0000043515 00000 n
0000042654 00000 n
0000043617 00000 n
0000040662 00000 n
0000042693 00000 n
0000000010 00000 n
0000039522 00000 n
0000040847 00000 n
0000042235 00000 n
0000042360 00000 n
0000042487 00000 n
0000042777 00000 n
0000042897 00000 n
0000043017 00000 n
0000043211 00000 n
0000043137 00000 n
0000043373 00000 n
0000039543 00000 n
0000040641 00000 n
0000042612 00000 n
trailer
<<
/Size 21
/Root 3 0 R
/Info 1 0 R
>>
startxref
43683
%%EOF
```

Het getal 43683 duidt de verschuiving aan van de cross-references tabel ten opzichte van het begin van het bestand.

We zullen het object 1 wijzigen: vooreerst zullen we een fout van Acrobat Distiller voor de (Franse) Macintosh versie verbeteren: de datum bevat 8-bit karakters (de ‘é’ in ‘Déc’) die Distiller verzuimd heeft te wijzigen naar de PDF codering. Het dialoogvenster met als doel bepaalde inlichtingen te vertonen, waaronder de datum van aanmaak van het document, zal een foutief karakter vertonen in plaats van de ‘é’. Dit 8-bit karakter moet veranderd worden door de code voor ‘é’ in PDF code: in PDF octale notatie, `\351`. We zullen dus een ‘é’ vervangen door viermaal `\351`: de verschuiving wordt +3.

Het betreft object 1, en het enige object die hierop volgt in het bestand is het object 3. De derde regel in de cross-reference tabel dient dus gewijzigd te worden (we rekenen vanaf 0: de regel

```
0000000000 65535 f
```

is dus de nul-regel. De eerste regel van de tabel is

```
0000043515 65535 n
```

deze stemt overeen met het object dat de informatie bevat. De derde regel is

```
0000043617 65535 n
```

en deze stemt overeen met het enige object dat zich bevindt na het object dat we zullen wijzigen (het nummer 43617 is inderdaad het hoogste in de tabel, het betreft hier wel degelijk het laatste object in het bestand). We zullen 3 optellen bij het getal 43617, alsook bij het getal 43683 op het einde van het bestand (de cross-reference tabel is ook met 3 verschoven). Het bestand wordt dus:

```
1 0 obj
<<
/CreationDate (Dim 25 D\351c 1994)
...
xref
0 21
0000000000 65535 f
0000043515 00000 n
0000042654 00000 n
0000043620 00000 n
...
startxref
43686
%%EOF
```

Hetzelfde geldt indien we enige extra informatie opnemen in object 1, we berekenen de uiteindelijke verschuiving en corrigeren vervolgens de cross-reference tabel en zijn pointer

```
1 0 obj
<<
/CreationDate (Dim 25 D\351c 1994)
/Producer (Acrobat Distiller 1.0 voor Macintosh)
/Author (Yannis Haralambous)
/Title (Voorbeeld van een PDF bestand)
/Subject (Conversie HyperLaTeX naar PDF)
/Creator (LaTeX, DVIHPS, en anderen...)
>>
endobj
...
xref
0 21
0000000000 65535 f
0000043515 00000 n
0000042654 00000 n
0000043762 00000 n
...
startxref
43828
%%EOF
```

Om de verbetering aan de cross-references tabel automatisch te laten gebeuren heeft de auteur het programma flex ontwikkeld, genaamd `recticrt`. Om een executable te verkrijgen voor de eigen omgeving dient de gebruiker vooreerst Flex (versie $\geq$ 2.4.6) op te starten voor `recticrt`, met als optie `-8` op de commandoregel.

```
flex -8 recticrt.lex
```

en vervolgens het uitvoerbestand van Flex te compileren met zijn C compilator (`lex.yy.c` onder UNIX of Macintosh, `lexyy.c` onder DOS enzovoort). Het gebruik van `recticrt` is zeer eenvoudig:

```
resticrt < oud bestand > nieuw bestand
```

Indien de cross-reference tabel correct is zal `resticrt` deze intakt laten.

Een andere courante wijziging bestaat uit het onzichtbaar maken van de omkadering van de actieve zones van de hypertext verbanden. Elk hypertext verband is gedefinieerd door een PDF object van het type

```
12 0 obj
<<
/Type /Annot
/Subtype /Link
/Rect [ 148 399 177 411 ]
/Border [ 1 1 1 ]
/Dest [ 4 0 R /FitH 607 ]
>>
endobj
```

De commando's `/Rect` en `/Dest` bepalen de coördinaten van de actieve begin- en eindzones. De parameters van `/Border` zijn:

1. de horizontale straal
2. de verticale straal
3. de lijndikte

De defaultwaardes zijn `1 1 1`. Door de derde parameter te wijzigen door 0 maakt men het kader onzichtbaar. Indien met oplet dat er geen spaties toevoegt of wegneemt wordt er ook geen verschuiving veroorzaakt. Het betreft dus een wijziging die geen aanpassing vereist van de cross-references tabel en zijn pointer.

De lezer kan de volledige beschrijving van het PDF formaat vinden in [1].

7 Beschikbaarheid

Adobe Acrobat Reader is beschikbaar op diverse public servers. Adobe Acrobat Distiller kan bekomen worden via Adobe Systems Inc.

DVIHPS (de gewijzigde versie van DVIPS) is — op het ogenblik dat dit artikel geschreven werd — in het stadium van β-test. De beschikbaarheid ervan (of de integratie van de functionaliteiten ervan in DVIPS, die in tegenstelling met TeX en METAFONT nog niet bevroren is) zullen aangekondigd worden via de gebruikelijke kanalen.

De utility `repere`, de overeenkomstige LaTeX style en de utility `recticrt` kunnen worden bekomen via de public server `ftp.ens.fr`, ze bevinden zich in de directory `/pub/tex/yannis/acrobat`. De lezer zal er tevens de voorbeelden beschreven in dit artikel kunnen vinden, alsook het artikel zelf (in Times, om de redenen vermeld in 3.2) onder de vorm `.tex, .bbl, .rep, .dvi, .ps` en `.pdf`.

Referenties

[1] Adobe Systems Inc, *Portable Document Format Reference Manual*. Addison Wesley, 1st edition, 1993.

[2] M. Goossens, 'LaTeX — HTML aller et retour', *Cahiers GUTenberg n° 19*, janvier 1995, 98–120.

[3] M. Goossens, 'Introduction pratique à SGML', *Cahiers GUTenberg n°19*, janvier 1995, 27–58.

[4] S. Rahtz, Hypertext marks in LaTeX. 1994.

[5] T.V. Raman, An audio view of TeX documents. In *13th annual TeX Users Group Meeting, Portland, 1992*, 1992.

Adobe™ Acrobat 2.0™ Beyond the bounds of paper*

Wiegert Tierie

Adobe Systems Benelux B.V., Europlaza, Hoogoorddreef 54a,
1101 BE Amsterdam Z.O., The Netherlands
wtierie@adobe.com

1 Introduction

Witness an exciting new development in the computer revolution. No longer content to *create* documents better, faster and cheaper, we've begun to look for ways to *use* documents better, faster and cheaper. We are in the midst of a shift from the paper trail to the information superhighway. While concepts like just-in-time and on-demand are familiar in referring to hard goods, we are just beginning to apply them to our truly liquid assets — information. Today, if you can move it, use it or manage it more efficiently than the next guy, you've got a competitive edge.

Consider the following:

- June 1993. The first copies of Acrobat are just rolling off the assembly line. Kenneth Grant and W. David Schwaderer wrote in the Adobe Acrobat Handbook, 'If the IRS used Acrobat, widely accessible income tax preparation forms would print on low-cost home printers, averting taxpayers' frantic trips to copy forms for last-minute filings.'
- Now roll ahead eight months: On February 21, 1994, *Business Week* reported, 'Say the Internal Revenue Service hasn't sent you an 8841 to request a deferred payment. Now, instead of scrambling to the nearest post office or IRS office, you can get digital copies of the forms sent directly to your home computer. CompuServe in Columbus, Ohio has arranged with the IRS and Adobe Systems to provide electronically any of the 450 federal tax forms over its on-line information service. First, subscribers to CompuServe®, which is owned by H&R Block Inc., have to download a copy of Adobe Acrobat software from the network — offered at no charge during an introduction period. Then, by searching for keywords, such as 'charitable gifts,' subscribers can find just the form they want and have it sent to their computer. Acrobat then reproduces the form on any printer — including dot-matrix models — almost as they appear on paper from the IRS.'
- President Clinton presented the FY1995 United States Budget to Congress in Acrobat Portable Document Format (PDF) on CD-ROM.
- Tandem Computer sends price lists, data sheets, product brochures and standard forms to field offices in PDF on CD, providing instant access to all product marketing and sales information.
- Over 100 World Wide Web sites offer PDF documents on their Web server in conjunction with the HTML documents. These documents include data sheets, product information, newspapers, forms, magazines etc.

These real-life scenarios illustrate the growing interest in electronic document software, products that let users create, view and print documents regardless of the computer, software or fonts used by the creator or available on the recipient's machine. Adobe Acrobat software strikes the best balance between screen and paper versions of documents, with its ability to navigate, link, search, crop and rotate; and with its PostScript™ language-based resolution independence and high-quality output.

2 What is Adobe Acrobat?

Acrobat software puts documents to work electronically. Anybody can create, use, store, share, view or print them. With any computer, any application and any printer.

Acrobat guarantees that the document you create is the document everybody sees. The Portable Document Format (PDF) has the flexibility to describe the content and appearance of documents created with virtually any application for virtually any computer or printer. Acrobat Exchange and Reader are powerful tools for accessing and managing information.

Adobe Acrobat is a family of products — Acrobat, Acrobat Pro, Acrobat for Workgroups and Acrobat Catalog — designed to bring the power of electronic documents to a broad spectrum of users. Each product contains the right collection of components for a specific user's needs (see 'Getting Started with Adobe Acrobat' later in this document).

The Acrobat Reader is the perfect tool for corporate and commercial publishers who distribute documents to large audiences. Acrobat Reader is included with all Adobe Acrobat products and can be freely distributed without charge to others. It allows recipients to view and print any PDF document they receive, and gives them access to all the

*The Acrobat CD-ROM Sampler was added with the shipping of this MAPS issue.

annotations, bookmarks and links that are part of the PDF file.

The Acrobat Exchange program gives users the power to exchange documents with other Acrobat users. They can create, view, collate, navigate and print PDF documents. PDF Writer creates PDF files from any application by 'printing' to a file when a user selects Print from the application's File menu. Included in Acrobat 2.0 is Acrobat Search, which provides full-text search capabilities for collections of PDF documents. Using Search software, users can quickly search indexed PDF documents for single words or terms, phrases and arbitrary character patterns specified with wildcard characters. They can also perform Boolean searches for documents that contain combinations of words and phrases.

Acrobat Distiller™ software lets individual and network users transform PostScript language files into PDF documents.

The Acrobat Catalog program creates full-text indexes for collections of PDF documents. For each word or term, the full-text index lists the documents and page numbers where the word or term is used. The Acrobat Search software uses full-text indexes to find words and terms in the documents quickly, without having to open the documents.

2.1 The flexibility to use any document, anywhere

Electronic document programs facilitate document distribution for users who are working on other computers, without requiring the recipient to have access to the applications used to create the documents. Not only the text, but formatting, fonts, page layout and graphics appear on-screen precisely as they would if printed on paper.

Electronic documents should be a natural extension of the existing work flow. To an everyday business user, that means working with any word processor, spreadsheet or presentation package and creating an electronic document as easily as selecting the Print command. It means adding value by easily combining elements from multiple applications into a single document, regardless of page orientation, fonts used and so on.

Commercial publishers, writers, graphic artists and engineers create more complex documents using applications and fonts that make use of industry-standard Adobe PostScript technology. Their work flow is different and demands an easy way to turn documents created with the PostScript language into shareable electronic documents, with the ability to fine-tune the file size and quality parameters.

Consumers look for the ability to view and print any document — not just documents they create, but those they receive from a multitude of sources — at the best quality possible. The process should fit seamlessly into the existing work flow and should accommodate any document.

Who needs electronic documents? Anybody for whom the communication, time value or management of information is critical.

Commercial publishers distribute materials electronically with all the visual enhancements that in the past required printed media.

Business organizations exchange proposals and reports electronically, enhanced with charts, diagrams, bold text, bullets and indentation. Price lists, data sheets, contracts and administrative forms are distributed days sooner. Technical documentation, blueprints and research are stored, managed and accessed more efficiently.

Information is central to most organizations, and making information more useful creates a competitive advantage.

2.2 More direct access to information

The power of information lies in the ability to use it. A fundamental benefit of electronic documents is enhanced access. Acrobat provides unparalleled facilities for finding and using information:

- Sophisticated full-text search allows users to find detailed information in thousands of documents — from the desktop, without assistance from reference librarians.
- Cross-document links add a new dimension to information access, linking disparate documents across the network and allowing information to be referenced and updated dynamically.
- Article threads automatically follow the story for ease of on-screen viewing.
- Bookmarks and links take the user to topics of interest.
- Thumbnails provide visual browsing and navigation.

2.3 Better ways to manage information

These days, it seems to be a given that hard disk use and network traffic will expand to exceed whatever capacity is allocated. Adobe Acrobat provides an opportunity to ease some of the strain of managing the varied collections of information that any organization must deal with:

- Documents can be indexed on existing networks without changing the way they are organized or stored.
- Document security controls access and reduces the need for multiple copies of documents in different locations.
- Multi-user notes allow workgroups to review and comment on material electronically.
- Compressed, platform-independent files can be archived on a central file server or distributed on a multi-platform CD-ROM.

2.4 An extensible architecture to customize and add value

Adapting electronic documents to an organization's work flow means tailoring products for unique environments. Acrobat Exchange 2.0 contains a complete set of Application Programming Interfaces (APIs) for creating plug-in modules, customizing the user interface and integrating

with other products. Leading systems integrators are incorporating Acrobat products into a wide variety of information systems.

2.5 More ways to integrate with other products

In today's resource-conscious market, businesses are looking for ways to create custom plug-and-play solutions from off-the-shelf components. In addition to the comprehensive APIs, Acrobat supports industry-standard interfaces to provide integration and customization features to the broadest set of customers, even nonprogrammers:

- Lotus Notes® F/X support to seamlessly share field-level information with today's most popular workgroup application. The information in PDF Document Info fields is used by Lotus Notes to organize their display in Views, giving users a powerful way to browse and find information across the network.
- DDE, OLE 2.0 and OLE automation support. As an OLE server, the Acrobat program integrates with important Windows™ applications.
- AppleEvent support to connect Acrobat with HyperCard®, AppleScript™, FileMaker®, Microsoft® Excel and many other popular Macintosh® applications.

2.6 A platform for evolution

The PostScript page description language is a de facto standard that is the foundation of the Acrobat program's Portable Document Format. Device and resolution-independent Adobe PostScript technology ensures that the most important investment — content — will migrate effortlessly into the future. Adobe published the Portable Document Format just as it did the PostScript language definition, to guarantee that information is not locked into a proprietary and transitory format. Choose any platform, monitor or printer today — or ten years from today — and Acrobat documents will take advantage of the benefits they offer.

3 Technical Overview

The goal of Adobe Acrobat is to reproduce any document (authored yesterday, today or tomorrow) as faithfully as possible, in a form that can easily be viewed, stored and distributed. Simple objective, many obstacles.

First, consider what's necessary to deliver on the goal:

- Create documents that handle any font or graphic, from applications that may no longer be marketed or may not yet be developed.
- Work with the hardware and software that users have or will have.
- Create small files.
- Perform well in networked workgroup environments.
- Assist users in navigating documents and finding the information they need.
- Help manage electronic content.
- Print to any printer — ImageWriter® to imagesetter.
- Provide an open, extensible and flexible file format

Now, a brief look at some of the problems and solutions in the areas of fonts, graphics, content independence and compression.

3.1 Platform coverage

Adobe Acrobat software is the only electronic document solution that works seamlessly in the DOS, Windows, Macintosh, and UNIX® environments. Because all Acrobat products are based on core code that describes a document's content independent of hardware platform or imaging model — and because the format is available as an open, published standard — users are assured that they will always have solutions tailored to their current computing environment.

3.2 The key to quality and longevity — content independence

The Adobe Acrobat program's file format is PostScript language-base — not a bitmap image of the page. Text is maintained as actual text characters in a specific font; graphics are maintained as lines and Bézier curves; images are maintained as monochrome, grayscale or color, just as they were in the original document.

Portable Document Format (PDF) is a published standard[1] and does not rely on QuickDraw™ or GDI.

Why is this important? The market moves too rapidly to track improvements to hardware and software in a timely way — device and resolution independence enable documents to be displayed or printed at the full resolution of next year's device. If you print to an 800 dpi laser printer, your text and graphics will print at 800 dpi. If you zoom in to 538%, the text and graphics will render at exactly that resolution and give you the best quality possible. In addition, a published file format ensures the openness of the solution and enables third-party vendors to embrace Adobe Acrobat and the PDF file format standard to deliver a wide range of solutions based on Adobe Acrobat.

Sure, flexible document handling covers common office documents, the stuff we churn out daily on our trusty word processors and empty into dumpsters weekly. But the flexibility to handle any document means legacy data and 'high-end' graphics, too. The PostScript language is the turf of both power graphics users and technical specialists using sophisticated equation fonts. Some kinds of documents can only be printed to Adobe PostScript output devices. Complex graphic artwork and images in Encapsulated PostScript (EPS) language format must be processed by a PostScript interpreter to reproduce at their full resolution.

How does Acrobat deal with PostScript language files?:

- 'fi' and 'fl' ligatures in PostScript Type 1 fonts

[1] Bienz, Tim and Cohn, Richard. *Portable Document Format Reference Manual.* Adobe Systems Incorporated/Addison-Wesley Publishing Company, 1993. ISBN 0-201-62628-4.

- Handling of PostScript Type 1 fonts included in the PostScript language file
- PostScript Level 2 files, even where PostScript Level 2 is not supported
- Search for text within an EPS graphic
- Print without loss of quality
- Compare the size of the original PostScript language file to the PDF document
- Recognition of PostScript page-size operators or commands; you shouldn't have to manually change the page size in the Page Setup dialog box each time you want to convert a PostScript language file that is a different size or orientation (Acrobat handles this task automatically)

How does Acrobat handle these experiments? Flawlessly. Not only are all characters in fonts correctly captured and displayed on Macintosh, Windows, DOS and UNIX systems, but they are not stored as bitmaps, so you can display and print them at any resolution. The following screen shot shows a zoomed-in view at 410% (Acrobat is not limited to discrete magnification percentages).

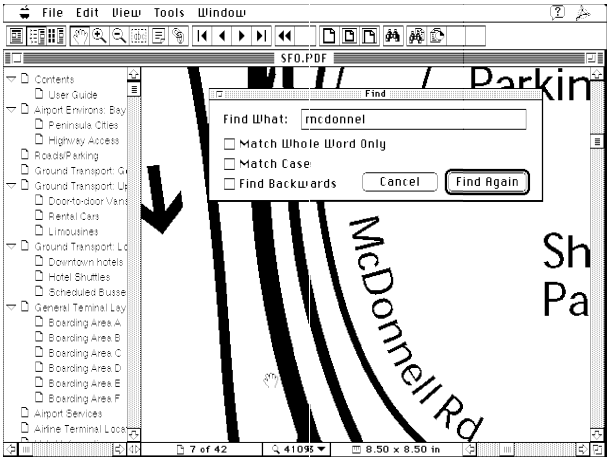


Figure 1: *How does Acrobat handle any document you throw at it? Flawlessly.*

3.3 Font handling

Fonts — their size, style and spacing — are the anchor of a document's appearance. Proper display of text is the toughest single problem in electronic document software.

Even users who have standardized on cross-platform applications and typefaces stumble over fonts when they try to share documents. Characters in fonts are referenced in platform-specific ways. Some characters are unique to a platform, such as 'fi' on the Macintosh. Acrobat handles font encoding intelligently to ensure that all characters are displayed properly regardless of platform. Further, Acrobat is the only electronic document program to offer full support for embedding both TrueType™ and Type 1 fonts. This document's Glossary contains useful information on font terminology.

There are three main approaches to representing fonts in electronic documents:

- Using local fonts
- Creating synthetic fonts (font substitution)
- Embedding the original font

Each approach involves tradeoffs among fidelity, performance and ease of use. Acrobat provides the full spectrum of options to meet the broadest set of user needs.

PDF files contain the names and metric information (called font descriptors) for all the fonts in the document. Adobe Type Manager™ (ATM™) or the native TrueType rasterizer rasterizes the requested font if it is available to the operating system; otherwise it creates substitute fonts based on font descriptor information in the PDF file. The layout, character spacing, graphics and general look of a PDF document are always preserved.

If a source document contains only the fonts installed with the Acrobat viewers, you don't have to worry about whether your readers have the fonts used in your document. But if your documents use other fonts, you can choose to let Acrobat create substitute fonts or to embed the fonts in the PDF file.

Font substitution — for smaller files

The Acrobat program's font substitution strategy creates synthetic fonts that closely simulate the appearance of unavailable fonts. It preserves exactly the spacing, alignment, stroke weights and other characteristics of the original font metric information stored within the PDF file. During the creation of the PDF file using the Acrobat PDF Writer or Distiller, the font metric information is extracted directly from the font itself (not the ATM font database, as people sometimes assume). Once the PDF has been created, the file itself is completely self-sufficient in terms of font metric information. The font metrics occupy only about 1K of space in the file, while embedding the entire font itself would use 25–30K.

ATM uses two special multiple master fonts to do font substitution: Adobe SansMM and Adobe SerifMM. Not available in application font menus, they were designed specifically to substitute for other fonts. Users can set preferences for substitution fonts to tune resource use versus quality. The default is both serif and sans serif, which uses both Adobe SerifMM and Adobe SansMM for font substitution.

Figure 2: *Acrobat uses the original font metrics and a multiple master font to preserve the characteristics of the original. On top is the original ITC Cheltenham® Condensed Bold font; below is the Acrobat substituted font, Adobe SerifMM.*

Font embedding — for absolute fidelity

What about decorative fonts, such as Adobe Wild Type™, which preserve the document look but lose their communication punch when replaced by substituted fonts? Or TrueType fonts, which vary between Macintosh and Windows, and aren't available for UNIX? By embedding the actual font program into the PDF file, Acrobat Exchange/Reader can display the font exactly the same way on Macintosh, Windows, UNIX and DOS.

The result? Type 1 or TrueType, simple or elaborate — not only are all characters in the font correctly captured and displayed on Macintosh, Windows, DOS, and UNIX systems, but they can be displayed and printed at any resolution (they are not stored as bitmaps). The following is a zoomed view at 410%.

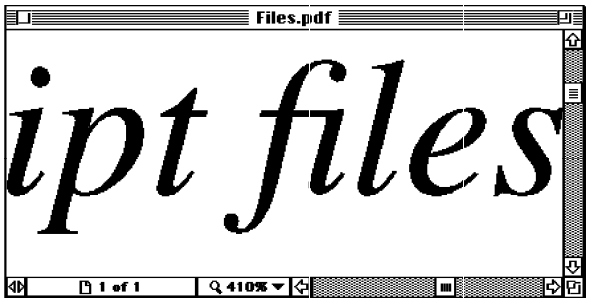


Figure 3: *Acrobat Exchange/Reader for Macintosh. Zoomed-in view (410%) of a document that was converted from a PostScript language file to PDF using Acrobat Distiller. Notice the 'fi' character and the high-quality appearance of the text, even at this odd magnification.*

As with font substitution, several options allow users to balance file size with fidelity. New in Acrobat 2.0, the Acrobat Distiller embeds font subsets — only the characters used in the document — to reduce file size and improve viewing performance.

It's useful to understand how Acrobat handles symbolic fonts. During conversion to PDF, the outlines or actual font itself for the symbolic characters are placed into the PDF file. These character outlines are then used for on-screen display in Acrobat Exchange or Reader. Below are 300-dpi scanned images of the printed output from Microsoft Word and Acrobat Exchange:

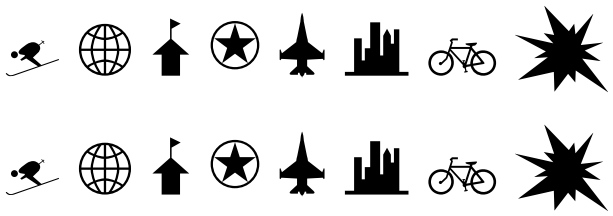


Figure 4: *Top, original Word document (printed at 300 dpi) and below, Acrobat Exchange document (printed at 300 dpi)*

Acrobat software properly displays symbolic fonts such as Carta™, because all the necessary font information (character shapes and metric information) are part of the PDF file and are guaranteed to be available for on-screen rendering and printing. (Acrobat does not rely on SuperATM™, which does not substitute for symbol fonts.)

Acrobat works with virtually any font; Type 1 and TrueType fonts can be embedded in PDF documents. Type 3 fonts are supported for printing and viewing — bitmaps are included in the PDF file.

Acrobat Exchange and Reader use the original TrueType font if the PDF file was created using the PDF Writer on that platform. The TrueType fonts in any original documents are converted by the driver to PostScript fonts (Type 1 or Type 3) if saved by a PostScript printer driver and converted to PDF by the Distiller. In this case, ATM does font substitution based on the font metric information in the PDF file. These fonts can be embedded by the Distiller.

In effect, you are embedding the Type 1 version of a TrueType font. (This is the case even if the named TrueType font is available to the operating system, because that font is designated as either a Type 1 or Type 3 font in the PDF file.)

3.4 File Size

Space is precious. A successful electronic document needs to be as small as possible within the confines of the creator's quality requirements. Several factors affect file size.

Compression results in smaller PDF files. Acrobat employs a variety of methods to tailor the file size to user needs:

- Text, graphics (line art) and indexed color images are compressed using the LZW method.
- Color/grayscale images are compressed using the JPEG (Joint Photographic Expert Group) method. This is a lossy compression method; data is removed from the image to produce the compression.
- Image downsampling removes pixel information to decrease resolution and reduce file size.
- Font subsets maintain perfect fidelity by embedding only the characters used in a document.
- Monochrome images are compressed using one of four methods. The default CCITT Group 4 method provides the greatest compression in most cases.
  CCITT Group 3 compresses monochrome bitmaps one row at a time and is used by most fax machines.
  LZW produces the best compression for images that contain repeating patterns.
  Run Length produces the best results for images that contain large areas of solid white or black.

Acrobat 2.0 defaults to a binary file format (retaining the option for ASCII) to further reduce file size and provide more reliable file transfer in certain environments.

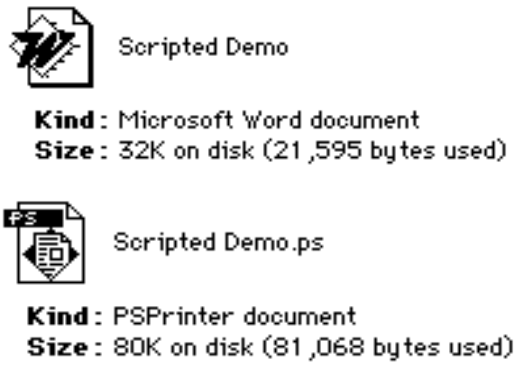


Figure 5: *A typical Microsoft Word file and the resulting intermediate PostScript file.*

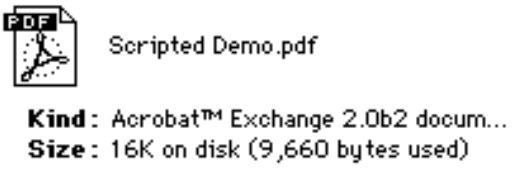


Figure 6: *The resulting PDF file is less than half the size (9,660/21,595 = 45%) of the native Word file.*

Where appropriate to the content, Acrobat 2.0's cross-document linking capability allows information to be broken up into separate files with links to related information.

4 Getting started with Adobe Acrobat

4.1 Acrobat Exchange (included in all Acrobat products)

The Acrobat Exchange program gives you the power to exchange documents with other Adobe Acrobat users. With it, you can create, view, collate, navigate and print PDF documents. PDF Writer creates PDF files from any application by 'printing' to a file when you select Print from the application's File menu. Included in Acrobat 2.0 is Acrobat Search, which provides full-text search capabilities for collections of PDF documents. Using the Search software, you can quickly search indexed PDF documents for single words or terms, phrases, or arbitrary character patterns specified with wildcard characters. You can also search for documents that contain combinations of words and phrases.

4.2 Acrobat Search (included with Acrobat 2.0)

Acrobat Search software gives you full-text search capabilities for collections of PDF documents that have been indexed with the Acrobat Catalog program. Acrobat Search uses full-text indexes to find words and terms in the documents quickly, without having to open the documents.

Using Acrobat Search, you can quickly find indexed PDF documents for single words or terms, phrases or arbitrary character patterns specified with wildcard characters. You can also perform Boolean searches for documents that contain combinations of words and phrases. Unlike most other full-text search products, Acrobat Search software can find words and terms in illustrations, graphs and formatted tables. It can even find words that are rotated or attached to a curved line.

4.3 Acrobat Distiller (included with Acrobat Pro and Acrobat for Workgroups)

The Acrobat Distiller program creates PDF files from virtually any document that has first been saved as a PostScript language file. Distiller is recommended for high-quality reproduction of EPS artwork, 24-bit images and documents that take advantage of features (such as blends) that are only available on Adobe PostScript printers.

4.4 Acrobat Catalog (included with Acrobat for Workgroups)

The Acrobat Catalog program creates full-text indexes for collections of PDF documents. A full-text index (also referred to as an 'inverted word list') is an alphabetized list of all the words and terms that are used in a collection of documents. You can, for example, build an index for all the PDF documents on a CD-ROM. You can also point Acrobat Catalog at your largest network file server and automatically build an on-line index for all the PDF documents. As documents are added, modified or removed, the index is updated incrementally to ensure that you always search against the files currently on the network. When used with indexes built for large PDF document collections, Acrobat Search enables users to search hundreds or thousands of documents in seconds.

5 Glossary

article threads – Documents in which text is in columns or otherwise separated and can be 'threaded' so that the reader can follow the flow. Acrobat Exchange automatically scrolls, centers the view and travels to additional pages in the thread.

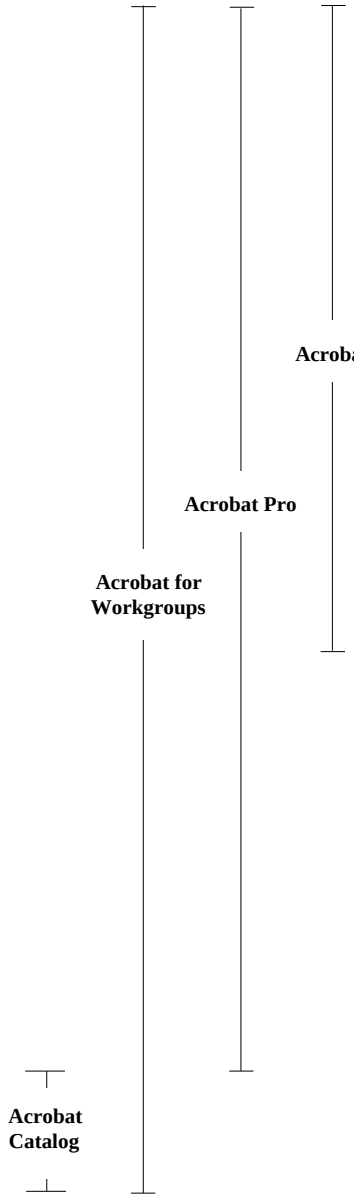
bookmarks – Used to mark parts of a document for quick access. Bookmarks are listed in the overview area of the Exchange window to locate hard-to-find or often-used information, or to create a custom outline of a document. Clicking a bookmark displays the location marked by that bookmark. Bookmarks can have the same actions as links (see that entry).

CCITT – An abbreviation for the International Coordinating Committee for Telephony and Telegraphy standards body.

CCITT Group 3 – A lossless compression method used by most fax machines that compresses monochrome bitmaps one row at a time.

CCITT Group 4 – A lossless compression method used by fax machines and applications that compresses groups of monochrome bitmaps.

downsampling – Deletes pixel information in an image to decrease resolution and reduce file size.



Use	If you	For
Reader	Consume information provided by others (without comment)	Browsing using links,bookmarks and thumbnails; printing
Exchange/PDF Writer Search	Review others' documents	Browsing and printing (as above) as well as adding notes, cropping, rotating, adding navigation
	Create and receive a wide variety of documents	"Printing" memos, reports or graphics directly from your application to shareable PDF files
	Collate and distribute documents from multiple platforms or application	"Electronically stapling" pages from many documents; drag-and-drop page placement
	Need instant access to detailed information in multiple documents	Performing sophisticated searches of PDF document indexes
Distiller	Create documents with today's page layout, graphics and image editing software	Maintaining the high quality of PostScript language graphics, images and Type 1 fonts
	Need to distribute documents created on mainframes to PC and workstation users	Translating legacy documents to shareable PDF files
	Need to maintain the high quality of PostScript language graphics, no matter what platform they're viewed on	The best resolution each computer can offer and the greatest flexibility in content and platform coverage
Catalog	Manage large volumes of of information on a network	Automatically building on-line indexes of all PDF documents

Figure 7: *An overview of the Adobe Acrobat product family*

Encapsulated PostScript (EPS) – An EPS file contains PostScript page description language information and maintains the full image quality across applications and output devices.

font encoding – Characters in fonts are referenced in different ways depending on the computer platform. The way Windows 3.1 finds a lowercase 'a' in a font is different from the way Macintosh or Sun™ workstations find the lowercase 'a' in the same font.

font formats – Include Type 1 fonts, Type 3 fonts, multiple master fonts, TrueType fonts.

font metrics – Specify character widths so applications can determine line lengths. There are two general types: integer metrics and fractional metrics. Font metrics also store kerning information. Kerning is an optional adjustment factor for the spacing between two characters. A positive kerning value moves characters apart; a negative value moves characters closer together.

font substitution – PDF files carry information about the fonts used in a document without actually including the font. Acrobat substitutes a multiple master font to maintain the look and feel of the original document.

full-text index – An alphabetical list of all the words and terms used in a collection of documents. Also referred to as an inverted word list.

JPEG (Joint Photographic Expert Group) – Provides significant compression of color and grayscale images. It is a lossy compression method; data is removed from the image to produce the compression.

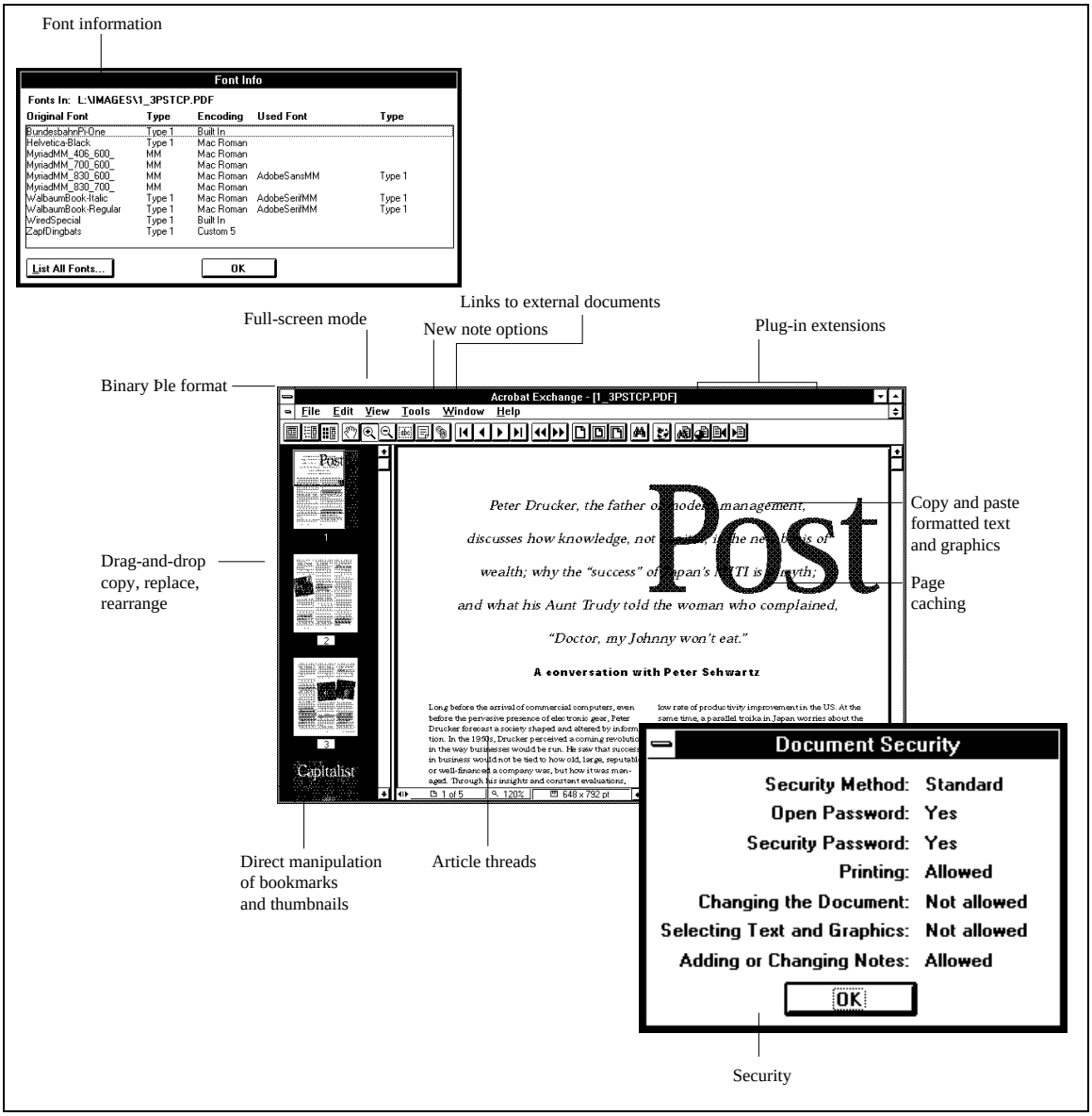


Figure 8: *Acrobat 2.0 at a glance.*

ligature – Two or more letters tied into a single character. The sequences 'fi' and 'fl', for example, form ligatures in most serif typefaces.

links – 'Hot' areas or text on a document page that the user clicks to travel to an associated destination. Links can go to another view or page in the document or another PDF document; open another file or program; or perform an arbitrary action defined by a plug-in.

LZW (Lempel-Ziv-Welch) – A lossless compression method that is useful for data containing repeating patterns.

multiple master fonts – Allow automatic copy fitting by starting with predetermined base designs and manipulating width, weight and optical scaling to create all other occurrences of the font.

page caching – Storing already viewed page images in memory to improve display performance.

Portable Document Format (PDF) – The key to cross-platform functionality of Adobe Acrobat products is a unique PostScript language-based file format. PDF is an open standard that Adobe Systems documents and publishes for use by software developers.

proximity – The Proximity search option changes the way found documents are assigned a relevance ranking. With the Proximity option selected, the closer the words are together in a document, the higher the relevance ranking.

Run Length Encoding – A compression method that produces the best results for images containing large areas of solid white or black.

sounds like – An Acrobat Search option that finds incorrect spellings of a search word. Searching for misspelling, for example, also finds mispelling.

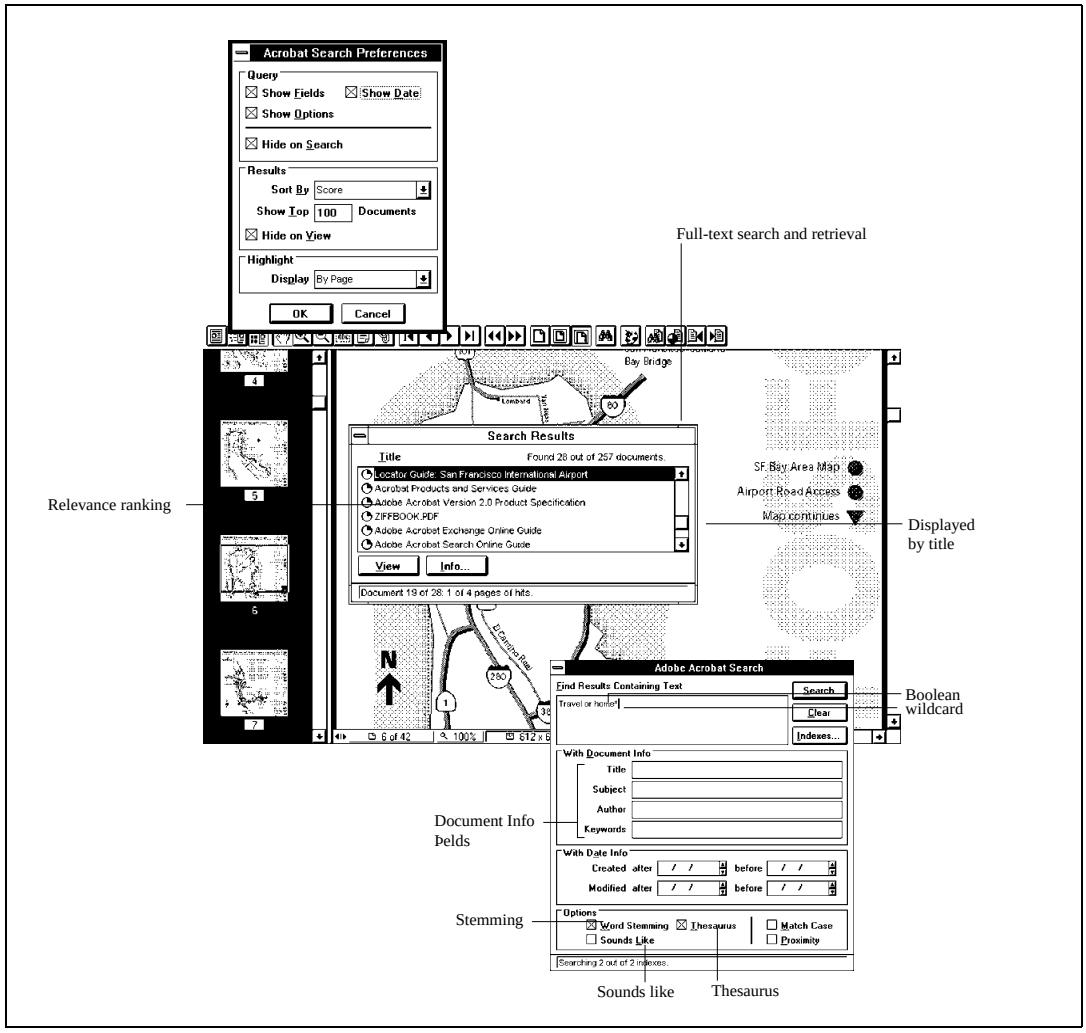


Figure 9: *Acrobat Search provides powerful data access.*

stopwords – Terms that are excluded from a full-text index. Common examples are articles, conjunctions and prepositions.

thesaurus – An Acrobat Search option that finds words that have the same meaning as the search word. Searching for crypt, for example, also finds mausoleum, sepulcher and tomb.

thumbnails – Miniature pages in the overview area of the Acrobat Exchange window that allow users to jump quickly to a page; adjust the view of the current page; and move, copy and replace pages in a PDF document.

TrueType fonts – Come in two forms on the Macintosh: bitmap and outline in one unit, or bitmap only in one unit. They are converted to Type 1 or Type 3 fonts by the PostScript printer driver when a PostScript language file is generated.

Type 1 fonts – Actually PostScript language programs, they were the first outline fonts. They are called outline fonts because each character is described mathematically as a collection of lines and curves that form the character's outline.

Type 3 fonts – Usually bitmap format instead of outline format. They are usually not hinted and tend to have lower quality than Type 1 fonts. Type 3 bitmaps are included in the PDF file. Type 3 fonts are PostScript fonts that do not work with ATM; they allow use of the full PostScript language, and ATM is not a full PostScript interpreter. Type 3 fonts are not hinted, resulting in poorer quality for screen display and when printing at small point sizes, and can exist in bitmap or outline form. They are useful for complex or ornamental fonts or logos, but they require more time for a printer to image.

wildcard characters – Characters used in a search to find all the words that contain a word fragment, or all the words and terms that match an arbitrary character pattern. The wildcard characters are the asterisk, which matches zero, one or more characters; and the question mark, which matches any one character.

word stemming – An Acrobat Search option that finds all the words with a common stem. With word stemming enabled, searching for *manager* also finds *manage*, *managed* and *managing*.

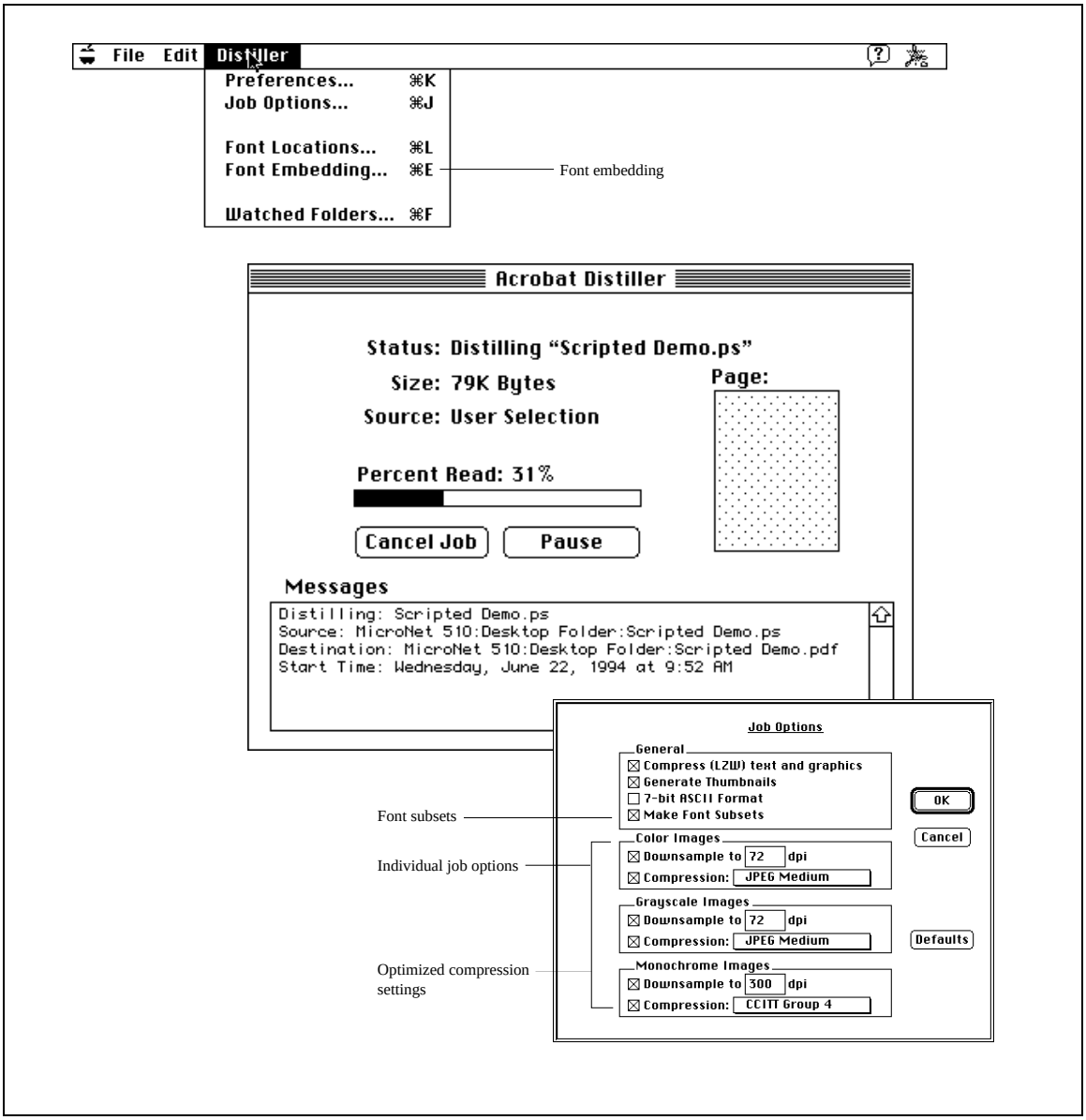


Figure 10: *Acrobat Distiller creates PDF files from virtually any document — created on DOS, Windows, Macintosh, UNIX or mainframe machines.*

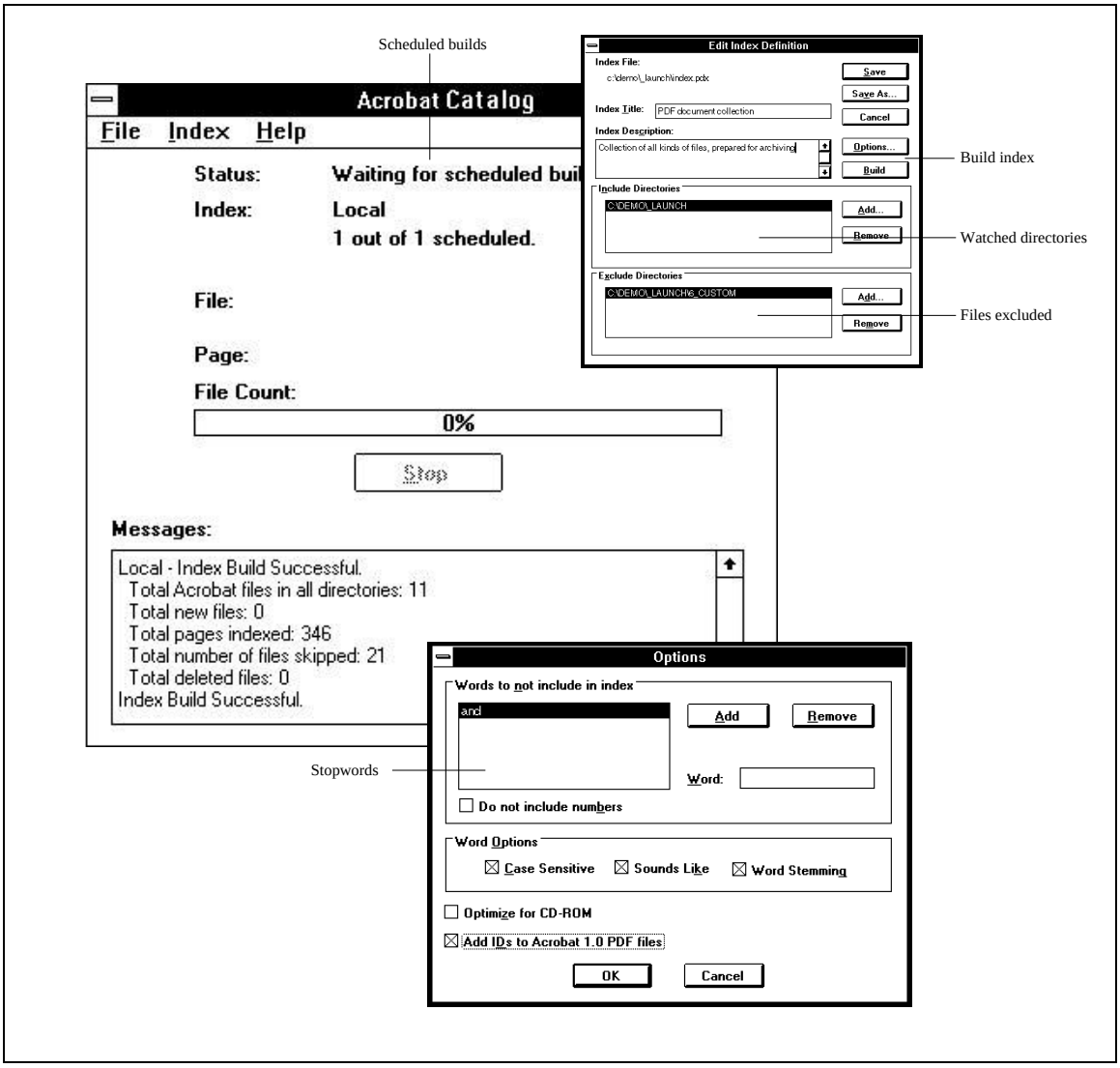


Figure 11: *Acrobat Catalog manages large volumes of information on a network.*

Conversion from WORD/WordPerfect to LaTeX*

Marion Neubauer

SFB 245
Psychologisches Institut, Universität Heidelberg
Hauptstr. 47–51
D-69117 Heidelberg
Marion.Neubauer@urz.uni-heidelberg.de

Abstract

Production of a large document with many contributors may require conversion of all submitted manuscripts into the same format, for example LaTeX. A large proportion of manuscripts are submitted in the formats of WORD and WordPerfect, two very popular word processing programs. I will discuss different approaches converting such files to LaTeX format.
First of all the differences between the word processors WORD and WordPerfect versus the document preparation system LaTeX will be explained, and problems encountered during text conversion into LaTeX will be discussed. The conversion can be done either by means of a seperate program (external conversion) or using macros, style sheets and a printer driver from within the word processors (internal conversion). Advantages and disadvantages of both methods for different types of text elements such as plain text, lists, tables and mathematical formulas will be discussed. This is followed by an overview of the conversion programs currently available.

Keywords: conversion, WORD, WinWord, WordPerfect, RTF

1 Introduction

First I would like to explain how I come to be working on the subject of this paper. I am employed at the University of Heidelberg and working in the special collaborative program, SFB for short, no. 245, called 'Language and Situation'. Psychologists and linguists at the Universities of Heidelberg and Mannheim and at the Institut für Deutsche Sprache in Mannheim are involved in this program. SFBs are supported by grants for pure research granted by the Deutsche Forschungsgemeinschaft (German Science Foundation) for an initial period of three years, after which a research report is produced and application is made for a grant for a further period of three years. This combined report and application for grant renewal comprised some 1000 pages and was supposed to be produced using LaTeX. The contributions, fiveteen in the current period, were written using various software programs including plain TeX, LaTeX, WORD, WORD for Windows and WordPerfect.

I will concentrate on conversion to LaTeX. Two of the programs described below also support conversion to plain TeX, although I will not go into greater detail.

Firstly, some terms: WORD refers to the program Microsoft WORD versions 5.0 and 5.5 running on the DOS operating system. Unless otherwise stated WinWord refers to WORD for Windows versions 2.0/6.0, WORD for DOS version 6.0 and also WORD on other operating systems. WP stands for WordPerfect for DOS versions 5.x. Most of the points concerning WinWord also apply to WordPerfect for Windows.

2 Overview

WORD, WinWord, and WP are word processors, in contrast to LaTeX, which is a document preparation system. These two concepts are generally understood to be partially or completely incompatible. This is actually the case with the versions available today.

I am assuming a familiarity with LaTeX and its markup language. 'Generic markup means adding information to the text indicating the logical components of a document such as paragraphs, headings and footnotes. The formatting (visual representation) associated with a component is decoupled from it's function (position) in the (hierarchical) structure of the text. LaTeX is, to a large extent, an example of a 'generic markup language' (Goossens, Mittelbach, Samarin: *The LaTeX Companion*, Addision Wesley, 1994, p. 7). For those who are not familiar with WORD I would like to give a brief description of the functions and the main differences between these two programs.

WORD is a so-called WYSIWYG program, i.e. what you see is what you get. This means that manipulated text ap-

*This paper was first published in the Conference Proceedings of the European TeX Conference, EUROTEX'94 in Gdansk. It was prepared within the context of the Sonderforschungsbereich 245, 'Spache und Situation', Univerität Heidelberg/Mannheim.

pears on the screen in its final form — for example, a word which is emphasized appears in italics on the screen. The manipulation itself, e.g. emphasizing a word, is done by marking the text and entering some key combinations or choosing the function from a menu. Character formatting in WORD involves specifying the name and size of a font, type style (italics, bold, bold italics, underlined, crossed out) and also sub- and superscript. WORD uses the IBM standard 256 character code table.

The record length of WORD text files is variable. A carriage return/line feed character is used only at the end of a paragraph. Standard LaTeX requires a carriage return character after every 500 characters, at the most at the size of `buf_size`.[1]

The file written by WORD — I will call it WORD text file — is very different from a LaTeX file. The file has a binary header, the text in ASCII and a binary trailer containing the formatting information in coded form. Pointers are used to apply the formatting information to the text.

WORD also stores formatting information outside the WORD text file, i.e. within an initialisation file of the WORD program, in style sheets and as part of the printer drivers used.

- Certain operating parameters of the WORD program can be changed at any time via the menu 'options' and apply to any text file loaded from then on. These parameters includes items such as the width of the tabulator.
- Style sheets are comparable to LaTeX style files. Within style sheets, layout functions are assigned to particular key stroke sequences such as page dimensions, layout of lists, etc. Key stroke combinations in style sheets can override standard definitions of WORD. For example, the key stroke combination Alt+I is normally used for italics, but in a custom style sheet it can be re-defined to format an item in a list.
- Printer drivers in WORD are not comparable with DVI drivers since they contain among other things the actual information about available fonts and their possible variations. WORD has a printer driver for every supported output device because the standard version can only use resident printer fonts. This is why changing printers can cause headaches.

WinWord stores almost all formatting information in the text file. Style sheets have a rather different function and WinWord uses the printer drivers of Microsoft Windows which use True Type fonts. Hence, the fonts are independent of the printer.

WP text files include all formatting information. WP use it's own printer drivers, it did not use the drivers from Windows.

3 Problems of text conversion

The following is a list of items which have to be considerd when converting files from a word processor to LaTeX.

- Record length of file,
- spacing:
  - horizontal spacing like blanks, tabulator stops, indentation,
  - vertical spacing, which is often realized with one or more blank lines,
- paragraph formatting: justified, flushleft, flushright, and centered text,
- type size and style (fonts),
- special characters:
  - diacritical symbols,
  - hyphens between compound words,
  - discretionary hyphens,
  - characters with a special meaning in LaTeX, mainly $, %, &, and #,
- headings, to be translated into LaTeX sectioning commands,
- lists and enumerations, which should be converted to the LaTeX list environments 'itemize', 'enumerate', and 'description',
- footnotes,
- mathematical formulas except sub- and superscripts, which are handled with a special font,
- tables, and last but not least
- indices.

In my opinion it is much more important that all characters of the original text are converted than to conserve all formatting information. A very good test of the text integrity is converting a percent sign because it indicates a comment in LaTeX, and % has to be converted to \% so that LaTeX will not get confused.

Personally, I like the conversion programs to handle more than just plain text, but due to the different typographic rules and the large differences of files created by word processors and LaTeX input files, it would be foolish to expect too much from a converter program.

4 Alternatives

All word processors discussed here can store a text in pure ASCII format. Once saved in this format, all formatting information is lost and has to be replaced in the form of LaTeX commands.

The alternative is to convert the formatting information into LaTeX commands by means of a program. I will distinguish two principally different ways, that are (a) 'internal' conversion, and (b) 'external' conversion.

4.1 Internal conversion

Internal conversion is carried out within the word processing program using macros, style sheets and (in the case of WORD) a special printer driver. The big advantage of the

[1] Common LaTeX versions today use a `buf_size` of 2000 characters.

internal conversion is the fact that all information about the text and formatting is accessible and usable. Remember, for example, the parameter of the options menu not stored in the text files.

Internal conversion is very simple and easy to use by people with experience using a word processor but with little or no knowledge of LaTeX. WORD, WP, and WinWord can then serve as a WYSIWYG editor to LaTeX.

The text is typed into WORD using macros and style sheets and then 'printed' to the hard disk using a 'LaTeX' printer driver. In WP and WinWord the text is stored using the normal storing procedure but with the format option 'DOS text format'. The resulting file is a complete LaTeX input file. Special text formatting needs can be met by modifying macros and style sheets. A small problem remains: line numbers, reported by LaTeX in case of an error, cannot be used because the lines are numbered differently within the WYSIWYG editors.

4.2 External conversion

External conversion is performed without the word processor by an external program. The text file is stored either in the text format of the word processor or in Microsoft's Rich Text Format (RTF). Many word processors including WORD, WinWord, and WP are able to write RTF files. RTF itself is a de facto standard. After the conversion, the file has to be edited using a text-oriented editor.

One problem remains: a conversion program can guess that a piece of text in 'Times Roman bold face 30 pt' is a heading, but without further information it cannot determine whether the heading introduces a section, a subsection or an appendix. Within WORD and WinWord the user has the opportunity of using a limited amount of pre-defined headings which can be converted to LaTeX sectioning commands.

5 Conversion Programs

5.1 Overview

Conversion of WORD text files:

- WORD2TeX (DOS), external
- WD2LaTeX (DOS), external
- WORD_TeX (DOS), internal

Conversion of WinWord 2.0 text files:

- WINW2LTX (DOS, Windows), internal

Conversion of files in RTF:

- RTF2TeX (UNIX), external
- RTF2LaTeX (UNIX, Mac), external
- RTFLaTeX (DOS), external

Conversion of WP text files:

- WP2LaTeX (DOS, UNIX), external
- WP2x (UNIX, DOS), external

5.2 Details

WORD2TeX, WD2LaTeX, and WP2x are external conversion programs for WORD or WordPerfect. WORD2TeX additionally includes a primitive version to perform internal conversions. The programs were written between 1989 and 1991 (see references for details), and as far as I know are no longer supported by the authors. They are useful mainly for plain text. Some of the special characters, like the German double s (ß), are converted incorrectly by some of the programs, especially those written by authors from English speaking countries (see Table 1).

In addition, contrary to what I have said above about external conversion not needing the word processor itself, WORD2TeX and WD2LaTeX produce better results when the text is stored by WORD with modified page dimensions before being converted.

WP2LaTeX is an external conversion program for WordPerfect written in C. It was updated in February 1994 and the author welcomes any comments about it. The older versions (first written in Pascal and later converted with Pascal-to-C) had a lot of errors, e.g. cutting footnotes after 256 characters. The new version has some improvements concerning formulas, but even they are not converted 100%.

WORD_TeX is an internal conversion program written for WORD version 5.0 only and converts documents to TeX and LaTeX. It has been used in my work to convert reports and applications. I wrote a modified style sheet for the authors which speeds up the conversion process considerably. For authors using WinWord I created a special WinWord style sheet. The text was stored in the 'Word for DOS' format and then converted using the same procedure as for the WORD 5.0 text files.

WORD_TeX is the best solution for internal conversion of text which has still to be typed in. It correctly preserves footnotes and paragraphs (including paragraph formatting) and also converts special characters, different type styles and sizes. Headings are converted into LaTeX sectioning commands and lists into the appropriate list environments. Tables are not converted: only the text of the table is transfered to the LaTeX input file. Sub- and superscripts from WORD text files result in the LaTeX command `\raisebox`. More complex mathematical formulas cannot be converted. WORD_TeX also offers half-automated conversion of quotation marks — the user has to confirm whether they are opening or closing quotation marks — and ligatures. The package is very well documented (18 pages, in German) and with little knowledge of WORD it can be modified to fulfil personal needs.

WINW2LTX is not really a converter but a style sheet for WinWord 2.0 to be used for editing LaTeX code. For example, choosing 'Bullets and Numbering' from the menu for a marked piece of text, the following commands are inserted `\begin{enumerate} \item ... \end{enumerate}` at the appropriate places. After the text has been stored under 'MS-DOS text with layout' it is ready to be processed by LaTeX. The program is still in the development stage. A yet it cannot convert formulas entered with the formula editor.

(As far as I know, there are two internal conversion programs for WP which were described in [1] and [3], but I have not been able to obtain copies of them. See references for details.)

RTF2TeX, RTF2LaTeX, and RTFLaTeX are external conversion programs, and use intermediate files in Microsoft's Rich Text Format (RTF). RTF is a very compact format and has a lot of similarities to TeX commands. It includes many positioning commands which cause problems during conversion. Some programs write redundant RTF code. This redundant code and the large number of positioning commands make LaTeX input files from these converters very difficult to read (and modify). Best results gives RTF-code from Macintoshis or WinWord 2.0b. The standard font for paragraphs should be times roman 12pt.

All three of these programs were written within the last two years and the range of texts that they can preserve is bigger t han with any other conversion program. As stated above, in most cases conversion of an RTF file only preserves the formatting information — information about text types is normally not stored in RTF. In addition, different type sizes are converted to LaTeX size commands, for example 40 pt will get `\LARGE`, 30 pt get `\Large` and so on. Because LaTeX has a limited amount of size commands a 1:1 conversion is not possible, but these programs give a close approximation.

The program RTFLaTeX has an option for producing a LaTeX file for LaTeX with NFSS. A style file belongs to the RTFLaTeX conversion package where serveral RTF commands are defined. After conversion, the style files have to be modified because only a subset of common RTF commands are defined. A bit of trial and error is needed to find out which commands are missing and what they should do.

The main advantage of RTF2LaTeX is the configuration of a character code map for all character codes between ASCII 128 and 255. A user-written configuration file can be used to convert WORD headings and other text styles, and an example is included in the distribution.

There is also a commercial converter from K-Talk Communication Inc. available, named 'Publishing Companion', price $249.00 (US). In an advertisement they promise to convert everything from the list given in the section 'Problems of text conversion', and even newspaper-style columns and mail merge.

6 Conclusions

1. Conversion of moderately complex documents remains a time-consuming job and the converters available today can only take over part of the work.
2. The word processors I have examined
   1. did not support a satisfactory standard markup (e.g. WORD), or
   2. the markup information is not available in the output file (e.g. RTF, except in WORD for Mac).

   These are the reasons why an automated external converter is not a viable proposition.
3. Using a word processor as editor in conjuction with an internal conversion program offers an acceptable solution. In the long term a change to a text-oriented editor like Emacs is still recommended.
4. For documents written in WORD I prefer the WORD_TeX converter, for conversion of large documents in WinWord or WP I choose RTF2LaTeX.

References

Most of the programs discussed are available from the CTAN Network (see *TUGboat* **14**(3):342–351, 1993). All programs are accompanied by source code. The location of the files on CTAN servers, e.g.
`ftp.dante.de:/tex-archive`,
or SIMTEL archive mirror, e.g.
`sun0.urz.uni-heidelberg.de:/pub/simtel`,
is given below.

[1] Hoover, A. Z. Using WordPerfect 5.0 to create TeX and LaTeX Documents. *TUGboat* **10**(4):549–558, 1989.

[2] Mintert, St. Recycling. Wie gut sind TeX(t)-Konverter? *ix* **8**:74–80, 1994.

[3] Modest, M. F. Using TeX and LaTeX with WordPerfect 5.0. *TUGboat* **10**(1):67–72, 1989.

[4] Microsoft Corporation. *Arbeiten mit Microsoft WORD für IBM Personal Computer und Kampatible.* Version 5, 1989.

[5] WORD2TeX (1989), Pascal, M. Lenz, Erlangen, Germany.
CTAN: `/support/word2tex`

[6] WD2LaTeX (1991), Pascal, Dr. Connor J. Thomas, Adelaide, Australia.
CTAN: `/systems/msdos/4alltex/disk05/wd2latex.arj`
SIMTEL: `/tex/wd2latex.zip`

[7] WORD_TeX (1992), Günter Partosch, Gießen, Germany.
CTAN: `/support/word_tex`

[8] WINW2LTX (1994), Allin Cottrell, Winston-Salem, USA.
CTAN: `/support/winw2ltx`

[9] RTF2TeX (1991), C, Robert Lupton, Princeton, USA.
CTAN: `/support/rtf2TeX`

[10] RTF2LaTeX Vers. 1.5 (1994), C, Erwin Wechtel, Wien, Austria.
CTAN: `/support/rtf2latex`

[11] RTFLaTeX Vers. 1.63 (1994), Pascal, Daniel Taupin, Orsay, France.
CTAN: `/support/rtflatex`

[12] WP$_2$LaTeX Vers. 3.0b (1994), C, Rob C. Houtepen, Eindhoven, The Netherlands.
CTAN: `/support/wp2latex`

[13] WP2x (1991), C, Raymond Chen, Berkeley, USA.
SIMTEL: `/tex/wp2x110.zip`

Table 1: *Conversion programs currently available*

Name	WORD2TeX		WD2LaTeX	WP2LaTeX	WP2x
Source format	WORD 5.0, 5.5		WORD 5.0, 5.5	WordPerfect	WordPerfect
Traget format	Standard LaTeX		Standard LaTeX	Standard LaTeX	Standard LaTeX
Conversion type	int.	ext.	external	external	external
Line breaking	+	+	+	+	+
Paragraph formatting[a]	−	−	−	−	+
Type style and size[b]	+	−	+	only style	+
Special characters[c]	+ (except &)	+ (execpt &)	+ (except ß)	+	+
Sectioning	+	−	−	+	−
Lists[f]	−	−	−	−	−
Footnotes	−	−	+	+	+
Mathematical formulas	−	−	−	−	−
Tables	−	−	−	−	−
Index	−	−	−	−	−

Name	WORD_TeX	RTFLaTeX	RTF2LaTeX
Source format	WORD 5.0, WinWord stored in WORD 5.0 format	All files stored in RTF	All files stored in RTF
Target format	TeX, Standard LaTeX, GERMAN.STY	Standard LaTeX, NFSS	Standard LaTeX
Conversion type	internal	external	external
Line breaking	+	+	+
Paragraph formatting[a]	+	+	+ (except flushleft)
Type style and size[b]	+	+	+
Special characters[c]	+	+	+[d]
Sectioning	+	−	+[e]
Lists[f]	+	−	−[e]
Footnotes	+	+	+
Mathematical formulas	−	+ (PostScript coded)	+
Tables	−	+	+
Index	−	+	+

+/−: Converted/not converted to the appropriate LaTeX command.

[a] These are justified, flushleft, flushright, and centered text.
[b] All character formatting type styles like bold face, italics, underlined, crossed out, sub- and superscripts.
[c] This includes diacritical characters as well as the different hyphens and also LaTeX specific characters.
[d] ASCII codes above 127 are configurable.
[e] Can be modified with a translation file for special WORD styles.
[f] This should be converted into the LaTeX list environments ‘itemize’, ‘enumerate’, and ‘description’.

Textures: zo goed als gezegd wordt?

Hans Renkema

Theologische Universiteit, Kampen
tel. 05202-24061

maart 1995

Als lezer van TUG-boat kwam ik reeds lange tijd de advertenties tegen van *Blue Sky Research* waarin hun commerciële TEX-implementatie TEXTURES wordt aangeprezen op punten van snelheid en gemak. Wie geen Macintosh heeft zou er alleen voor dit programma een kopen.

De redactie van onze MAPS regelde een recensie-exemplaar om de advertentiebeweringen op hun waarheidsgehalte te onderzoeken. Een Macintosh hoefde ik niet te kopen daar ik reeds de eigenaar ben van een Quadra 610.

Van *Blue Sky* ontving ik versie 1.6.3 met een keurig uitgevoerd handboek. De installatie verloopt zonder problemen. Het TEXen van bijgeleverde voorbeeldfiles eveneens. Het vervangen van het oude LATEX 2.09 format door het nieuwe van LATEX2_ε gaat eveneens probleemloos. Het nieuwe format kan met Virtex (het Initex van TEXTURES) snel gegenereerd worden. Van PSNFSS is er een speciale versie voor TEXTURES beschikbaar.

De eerste zaak waarin ik geïnteresseerd ben is de snelheid. Is TEXTURES werkelijk zo snel als er beweerd wordt? En hoe dat te meten? Het heeft weinig zin om alleen absolute tijden te geven. Dat verschilt voor elke type Macintosh. Beter lijkt het me de snelheid te vergelijken met die van bijvoorbeeld OzTEX, de veelgebruikte shareware implementatie van Andrew Trevorrow. Momenteel gebruik ik OzTEX 1.8.[1] Een forse LATEX-file van 500kb werd aangemaakt, hoofdzakelijke bestaande uit platte tekst. OzTEX deed er 143 seconden over en het bleken 137 pagina's te zijn. TEXTURES had daar slechts 95 seconden voor nodig. Gesteld kan dus worden dat TEXTURES dus 1.5 keer zo snel is als OzTEX. Men kan zich afvragen of de tijdwinst die men boekt in het geheel van de produktie van een document zo spectaculair is. Ik denk van niet. Wel is het zo dat het programma lekker snel aanvoelt, een niet onbelangrijk gegeven.

Het is echter niet alleen de snelheid waardoor het programma zich onderscheidt. TEXTURES is in de zgn. *'Flash'* mode in staat om de tekst die in het editorvenster wordt ingegeven tegelijkertijd te zetten en het resultaat even later in een venster ernaast te tonen. Dit is een buitengewoon handige mogelijkheid wanneer men ingewikkelde formules aan het zetten is en men snel het resultaat van ingegeven correcties ziet. Hetzelfde geldt als men bezig is met een ingewikkelde paginalayout of met de eindcorrectie van een document. Vooral als men met plain TEX werkt gaat dat erg snel. Bij LATEX-files is er meer oponthoud.[2]

Het programma moet een genoegen zijn voor mensen die niet goed uit de voeten kunnen met tfm- en pk-files. *Blue Sky* heeft alle CM-fonts van Knuth omgezet in een PostScript versie.[3] De tfm-files zijn in het programma zelf geïncorporeerd. De PostScript fonts staan in de map 'Lettertypen' in de systeemmap. Ook de gangbare fonts aanwezig in PostScript printers zijn zo in het programma ingebouwd. Gevolg is dat TEXTURES veel minder ruimte op de harde schijf inneemt dan een klassieke implementatie. Met vier à vijf megabyte komt men een heel eind. Verder ziet de TEXTURES-folder er een stuk minder ingewikkeld uit dan een klassieke TEX-folder, zeker wanneer men alle hulpprogramma's, tools en voorbeeldfiles eruit laat. Vandaar dat TEXTURES een goede keus lijkt voor mensen die met TEX beginnen.

Anders ligt dat voor de gevorderde TEX-gebruiker. Wil hij bestaande files met TEXTURES gebruiken, dan komt hij al gauw problemen tegen. Bijvoorbeeld wanneer men fonts gebruikt die niet in TEXTURES zijn ingebouwd. Te denken valt dan aan WASY fonts, de Griekse fonts, andere PostScript fonts en fonts van eigen makelij. Weliswaar worden er wel Tools verstrekt om daarvan speciale metrische files te vervaardigen die TEXTURES kan gebruiken, maar echt simpel gaat dat niet. Nog moeilijker en kostbaarder wordt dat als men bestaande bitmap-fonts wil gebruiken.[4]

Een ander punt is dat voor het gebruik van de PostScript fonts van TEXTURES het gebruik van Adobe's ATM noodzakelijk is. Ik ben daar geen liefhebber van. Het neemt

[1] Deze versie is wat langzamer is dan 1.7 omdat Trevorrow een andere compiler gebruikt in verband met de komende release voor de Power Macintosh.

[2] Uiteraard wordt dit korter en dus minder storend naarmate de computer sneller is. Inmiddels is er van TEXTURES een versie verschenen (1.7) voor de Power PC waarvan de snelheid aanmerkelijk groter moet zijn.

[3] Ook zijn True Type versies te verkrijgen. Helaas ontbreekt cmcsc. Een nadeel is dat de fonts niet zijn aan te passen aan de printer via de waarde van de 'blacker' in de mode.mf file.

[4] Weliswaar levert *Blue Sky* programma's voor het omzetten van pk-fonts naar Macintosh-fonts, maar daar moet wel de kostbare MPW SHELL voor gebruikt worden.

een aanzienlijk stuk geheugen in beslag en het vertraagt de schermafhandeling. Bitmaps zijn sneller.

Voor gevorderde TEX-gebruikers zou het programma een stuk aantrekkelijker worden als *Blue Sky* een versie zou uitbrengen waarbij hun *'Engine'* zou werken in een klassiek environment, dus met tfm-folder en pk-folder etc. Ook de output zou rechtstreeks de klassieke dvi-file moeten zijn, wat de uitwisselbaarheid ten goede komt.[5] Qua gebruik van PostScript mogelijkheden doet TEXTURES weinig onder voor DVIPS.

Nog twee andere punten van kritiek. Het eerste betreft de editor. Deze is van de simpelste soort en eigenlijk niet om aan te wennen als men bijvoorbeeld met de shareware editor *'Alpha'* heeft gewerkt.[6] Een tweede punt van kritiek betreft de viewer. Deze mist de scrollmogelijkheden van de OzTEX-viewer, wat onontbeerlijk is voor kleinere schermen.

Tenslotte. Men kan bewondering hebben voor het team van Blue Sky dat er in geslaagd is TEX zo te implementeren dat de resultaten binnen de kortste tijd zichtbaar zijn. Dat is een grote aanwinst, vooral bij moeilijke teksten die veel ingeefcorrecties nodig hebben alvorens men het gewenste resultaat bereikt. In dat geval is de aanschaf van TEXTURES het overwegen waard, ook al moet daar een behoorlijk bedrag voor worden betaald. Dat laatste kan echter ook een bezwaar betekenen voor hen die TEX sinds jaar en dag uit de Public Domain sfeer kennen of gewend zijn slechts lage shareware bijdragen te betalen. Niettemin zou voor hen een gestripte versie [7] door zijn snelheid qua zetten en viewen heel aantrekkelijk kunnen zijn.

[5] Nu kan men alleen via een omweg dvi-files genereren.

[6] Wel begrijp ik uit de folder voor de versie 1.7 dat de editor verbeteringen heeft ondergaan.

[7] Dus zonder editor, maar te koppelen aan de editor naar keuze en functionerend in het klassieke environment met tfm- en pk-files.

Scientific Word / Workplace 2.0.1.*

What's new?

Jan Krugers

Gagelweg 3, 4651 VL Steenbergen
01670-64422

April 2nd, 1995

Scientific Word version 1.x from TCI Software Research has gotten improvements and an offspring. The improvements are not only better support for different LaTeX styles and dialects but also more user conveniences and better stability. The offspring is **Scientific Workplace**, a similar Windows software package under Windows 3.1. But besides all the Scientific Word functions it also has a built-in Maple Version V interface and many Maple symbolic calculation functions. Not to forget that version 2.0.1. works fine under OS/2 Warp with all its speed and data exchange advantages.

1 Additions

Since the introduction of Scientific Word in 1992 LaTeX2$_\varepsilon$ has been released. This new version of Scientific Word supports the new version of LaTeX. A major benefit is the New Font Selection Scheme. This allows full use of Truetype fonts.

Another use is the graphically oriented style editor. The layout of a document can be graphicallydetermined. This includes placing of headers, footnotes, margin notes, font, fontsize and many other items.

Many documents are made in $\mathcal{A}MS$-LaTeX. Scientific Word can create documents in this format. This is further supported by the rather extended $\mathcal{A}MS$symbols set. As before these symbols are easily selected from symbol panels.

However these symbol panels can be edited by the user. Not only can the most used symbols be placed in a cache bar as before, but other often used symbols can be placed on the symbol panel where it is the easiest for the typist.

Another handy feature for the typist is automatic substitution that allows the keying of part of a mathematical expression, the program then adds automatically the rest. You can determine your own substitutions. The same tactics apply to the accented characters. They can be inputted from the keyboard, a symbol panel or from the cache bar.

Not only supports Scientific Word $\mathcal{A}MS$-LaTeX but also REVTeX as defined by American Physical Society, Optical Society of America and the American Institute of Physics. As these institutes do not allow others to distribute their macros the user has to download them from one of their bulletin boards. But they do work under Scientific Word.

BibTeX is included to allow easy generation of bibliographies. Indexes can be made much faster. This is important for large documents. These are also better supported. You can make a file calling parts of a large document. Each part can be handled separately or for the final print as part of the complete book. Useful is the new compile facility. This allows to precompile a document before printing or previewing. As before Scientific Word senses when a document has changed and will then insist on a new compile.

Graphics can be imported and placed according to different TeX possibilities in the document. The following graphics formats are supported:
WINDOWS DIB (bmp), COREL DRAW (cdr), METAFILE (cgm), CLEAR TEXT CGM (ctm), MICROGRAFX DRAWING (drw), AUTOCAD INTERCHANGE (dxf) ADOBE ILLUSTRATOR (ai), ENCAPSULATED POSTSCRIPT (eps), HP GRAPHICS (hgl) (plt), LOTUS PICTURE (pic), MACINTOSH PICT (pct) WINDOWS METAFILE (wmf), GRAPHICS INTERCHANGE (gif), JPEG (jpg), PAINTBRUSH (pcx), TAG IMAGE (tif), WORDPERFECT (wpg) DIGITAL RESEARCH (img), PAINT (msp) IBM IOCA, GRAPHICS (ica), SUN RASTER (ras), ASCII (txt), MAC PAINT (mac), PHOTOSHOP (psd) CALS (cal), LASERVIEW (lv), XBITMAP (xbm), PIXMAP (xpm), KODAK PHOTO CD (pcd), HALO (cut), ICON (ico), AMIGA (iff), CLIPBOARD (clp)

Also you can import files in RTF format (from e.g. Word).

Additions can also be found due to Windows improvements released by Microsoft. Scientific Word is delivered with Win32 speeding up all your applications which can use 32 bits. Scientific Word itself uses it for the previewer. You have the choice of a 16bit or a 32bit one. The latter one works really fast under Win-OS/2. Another released Windows features supported are drag and drop of text in one document.

An addition, which you may wonder why it was not there in the first place, is the import of the contents of a file. You can import another LaTeX file and give it the style of the document you are working with.

*A Hypertext Scientific Word / Workplace demo diskette is included with this MAPS.

2 Improvements

Tables can be made fully in the text mode, no TEX is visible anymore and the table is presented fully WYSIWYG while you are composing it. It resembles somewhat the ease with which you always made very large matrices and filled them with complex expressions.

Predefined print styles have been improved and many have been added. Some of them are modifiable thru the graphical style editor. All available styles are listed in the hypertext demo of Scientific Word.

Lists can still be made included nested ones. You can assign your own symbol as list marker.

Customization of the screen, how the different document parts are displayed, has been made more flexible. You can define a default appearance.

Cross references are fully automatically resolved. You do not have to determine anymore the number of passes the compiler should make.

3 Still there

Good things do not die. Hence you still input text and mathematical formulas in WYSIWYG mode. However they are stored internally in a LaTEX format. This can be made visible in the TEX field box via the clipboard. The new Scientific Word specific file manager allows a view of the whole document in ASCII. In inputting formulas you indicate the mathematical attribute (like subscript) and the program determines font size and placement. The placement depends on how you put the formula: in-line or centered by itself on a line (display type). Also the font size changes between these two. In version 2.0.1. you can work now with multiline display types and cross reference to each line in the multiline display. Still there is the facility to record the keystrokes for a complex mathematical formula and store this as a 'fragment'. Fragments can be replayed (CTRL+given name) and the formula adjusts itself to in-line or display type.

Still there is the interesting display of text and mathematics. Mathematics is displayed in another color so you can see if sine is meant as a word or as an expression. What has been added in composing mathematical formulas are spaces. Although not in the TEX specification this appeared to be necessary. For example after the italics in a mathematical formula you need some extra space in order for the upright character not to cross thru to the top of the previous italic character. There is a choice of different EM spaces. You select them from a symbol display. Greek and Mathematics symbols can also be selected from two other keyboards.

Of course the DVI printer drivers are still there. You can print, rather fast, to laser — postscript — inkjet and matrix printers. The document can still be spell checked in many different languages like Danish, French, German, Swiss German, etc. There is still the separate treatment of front matter (author, abstract,. . .) and the listing of document information like dates, typist. . .

4 Scientific Workplace

This offspring contains the EasyMath technology of TCI Software Research and a part of Maple V.

The EasyMath technology allows you to write the same formula in different ways. Obviously there always remains ways which will not be understood by the program but the known ways are supported. This approach is needed when one wants a formula to shine not only on paper but also wants to shine as the input to the symbolic calculation mechanism of Maple V. Maple has to understand the meaning of the formula. It then calculates the result and puts this preceded by an equal sign into the document. The document is immediately reformatted by Scientific Workplace to give the proper space to the result. The calculations are done on the highlighted part of a formula. This makes it possible to calculate on part of a formula.

The following classes of Maple functions are supported:

- EVALUATE (NUMERICALLY)
- SIMPLIFY
- COMBINE: exponentials, logs, powers, trigfunctions
- FACTOR
- EXPAND
- PLOT 2D: rectangular, polar, implicit, parametric, conformal, gradient, vector field, ODE, phase place
- PLOT 3D: rectangular, cylindrical, spherical, implicit, tube, gradient, vector field
- SOLVE (ODE): exact (Laplace), numeric (series), integer, recursion
- CALCULUS: integrate by parts, change variable, partial fractions, approximate integral, plot approx., integral, find extrema, iterate implicite differentiation
- SIMPLEX: minimizedual, feasible, maximize, standardize
- STATISTICS: fit curve to data, random numbers, mean, median, mode, correlation, covariance, mean, deviation, moment, quantile standard deviation, variance
- CHECK EQUALITY
- DEFINE: new, undefine, show, clear, save, restore, define maple name
- SERIES
- VECTOR CALCULUS: Jacobian, Hessian, Scalar potential, Vector Potential
- MATRICES: adjugate, concatenate, characteristic, column basis, condition number, definiteness test, determinant, eigenvalues eigenvectors, fill matrix, fraction free Gaussian, Hermitian transpose, inverse, Jordan from, minimum polynomials
- POLYNOMIALS: collect, divide, partial fractions, roots, sort, companion matrix

This Maple kernel which is included in Scientific Workplace operates in the background and is called via a menu item. The generated plots can be included in the document. This gives you a full word processor for marvelous printouts of formulas with their results and their graphical representation.

Scientific WorkPlace*

een eerste indruk. . .

C.M. Fortuin

Hogeschool Gelderland,
Ruitenberglaan 26, 6826 CC Arnhem

Abstract

Het pakket SCIENTIFIC WORKPLACE. versie 2.0(SWP), vroeger *Scientific Word*, is een uitgebreide versie van een 'WhatYouSeeIsWhatYouGet' tekstverwerker gebaseerd op LaTeX2$_\varepsilon$ en werkend onder Microsoft Windows. De uitbreiding bestaat uit een deel van een 'Computer Algebra' pakket, de kernel van Maple. Verder is er een programma om toetsen samen te stellen, uit databestanden met opgaven. Deze voorlopige test van versie 2.0 is uitgevoerd op een IBM compatibele PC, met een 486 processor en 8MB intern geheugen, door een persoon met LaTeX2.09 en Maple ervaring, maar zonder kennis van MSWindows ('waar is dat nou voor nodig. . .') of SCIENTIFIC WORKPLACE. Voor het installeren en testen zijn twee volle dagen gebruikt. Dit verslag geeft vanzelfsprekend slechts een eerste indruk van het pakket.

1 Installatie

Het installeren van het volledige SWP pakket vraagt nogal wat van de computer. Het installatieprogramma waarschuwt daarom als het geen 30MB ruimte op de 'disk' vindt, maar beziet alleen de 'disk' waar vandaan het (onder Windows) wordt gestart. Zo kon de beschikbare geheugenruimte — door een 'ongelukkige' partitie — niet volledig worden benut. Helaas bleek geen rekening te zijn gehouden met de benodigde 'uitpakruimte': 35MB vrij geheugen is echt nodig voor het totale pakket; wie al het pakket Maple heeft staan kan met minder toe. Merkwaardig was de wispelturigheid van het installatieprogramma (of Windows?), doordat soms een boodschap kwam: 'onvoldoende geheugen' (niet dus), soms 'algemene beschermingsfout' (waartegen?). Gewoon volhouden bleek een oplossing te bieden. Het later apart installeren van het Maple deel, via het 'zelfkies'menu, bleek niet naar behoren te werken: de keuze werd volledig genegeerd, en *alles* werd opnieuw geïnstalleerd. Het installatieprogramma waarschuwde ook met betrekking tot de 'enhanced 386 mode' en gaf als minimale grootte van het virtuele geheugen 16MB (in verband met het afdrukken moet het virtuele geheugen ('swapfile')). Windows protesteerde echter tegen die keuze(?), en hield het op 15,3MB.

2 Aanmaken

Is het pakket geïnstalleerd, dan wordt een eerste echte test uitgevoerd op een meegeleverd bestand 'checkout.tex': dat werd keurig ge'compile'erd, ge'preview'd en ge'print' zoals het hoort. Is het bestand in de 'editor' gelezen, dan wordt de tekst zichtbaar in een Times Roman lettertype, maar worden de (LaTeX) commando's onderdrukt. De verschillende soorten tekst worden in een andere kleur, lettertype of lettergrootte getoond, zodat de structuur van de tekst duidelijk is. Dat geldt ook voor de wiskunde tekst, die rood wordt afgebeeld. Wel zijn aanwijzingen als vskip, enz. als tekst in grijze hokken te zien. Deze presentatie doet heel plezierig aan.

Om tekst toe te voegen moet veelvuldig gebruik worden gemaakt van TeX comando's, die met behulp van icoontjes en hulpvensters beschikbaar zijn. Wie gewend is aan het tijpen van commando's zal dat niet altijd even plezierig vinden. Wie Windows gewend is en kennis maakt met LaTeX zal daar anders tegenover staan.

3 LaTeX

Het inlezen van een 'bestaand' LaTeX bestand gaf onverwachte effekten: SWP negeert het vooraf-deel, vanaf

```
\documentstyle [dutch,11pt]{artikel}
```

tot en met

```
\begin {document}
```

in het bijzonder de documentstijl en de aanwijzingen voor de tekstbreedte en hoogte, maar ook de daar gedefinieerde macro's, waardoor een andere opmaak tevoorschijn kwam. Het moet toch mogelijk zijn de met SWP aangemaakte bestanden enerzijds, en anderzijds de bestanden gemaakt volgens de LaTeX conventie, door elkaar te gebruiken (door een wat andere inrichting van SWP). Een andere verrassing was het vinden van de tekst: 'paragraph' waar het document '`\paragraph*`' had staan: om een of andere reden is de `*`-optie in SWP niet geïmplementeerd.

Het pakket ondersteunt helaas het nederlands niet. Zo is er geen nederlands afbreekpatroon bestand ('NLHYPHEN'). Weliswaar is 'Babel' aanwezig maar ontbrak 'dutch'. Wel zijn er veel STY-bestanden, zo'n 100 verschillende. Verder

*Importeur is: Technical Marketing Consulting, Gagelweg 3, 4651 DC Steenbergen, Nederland; telefoon: 01670-64422.

miste ik in de 'editor' de mogelijkheid om 'gewone' TEX commando's toe te voegen; het lukte mij slechts via een ASCII tekst-verwerker.

4 Grafisch

Grafische bestanden van allerlei 'format' worden ondersteund:

- Computer graphics metafiles (CGM)
- Encapsulated PostScript Files (EPS)
- HP graphics language files (HGL)
- Micrografx Designer/Draw files (DRW)
- AutoCAD drawing interchange format (DXF)
- Tag image file format (TIF)
- Windows bitmaps (BMP)
- Macintosh PICT files (PCT)
- Corel Draw files (CDR)
- Windows metafiles (WMF)

Genoemde grafische bestanden kunnen in SWP worden gelezen en in een 'box' gezet. In goede LATEX traditie kan die verschillend worden gezet: in de regel, tussen de regels of als 'float'. Dat allemaal is niet door mij getest; kennelijk werkt de omzetting van de door Maple gegeneerde grafische bestanden uitstekend.

Het is voor mij de vraag hoe bijvoorbeeld een grafisch TEX programma als PICTEX, of het programma TexCAD, zou kunnen worden gebruikt binnen SWP. Dat lijkt nu slechts mogelijk via het maken van eigen stijlbestanden waarin de benodigde bestanden worden opgeroepen; verder ondersteuning is er niet. SWP lijkt erop gericht dat figuren worden gegeneerd met programmaas 'van buiten' (uitgezonderd dus Maple).

5 Maple

Werken met tekst én Maple gaat verrassend eenvoudig. Zo kunnen we een wiskundig probleem vanuit de 'editor' laten uitwerken door Maple. Bijvoorbeeld kan $(x+y)(x-y)$ worden uitgewerkt, de figuur ervan geplot en 'automatisch' worden ingevoegd in de bestaande tekst: $(x+y)(x-y) = x^2 - y^2 =$

Plot in .WMF bestand kan buiten Windows niet getoond worden, uitgezonderd de afmetingen van de framebox.

Daartoe is $(x+y)(x-y)$ in een blok gezet, Maple aangeklikt, en uit het menu respectievelijk expand en plot3d gekozen. De figuur wordt door SWP verwerkt via een 'special' TEXmacro \FRAME in de tekst opgenomen. Dat is prettig werken! Het lastigst was voor mij het herschalen van de figuur. Natuurlijk kan de figuur van onderschrift worden voorzien, op de gebruikelijke wijze.

Nog wat klein spul met Maple: $\int \cos x dx = \sin x$, en $\frac{1}{(x-1)(x-2)} = -\frac{1}{x-1} + \frac{1}{x-2}$

6 Afdrukken

Bij het installeren wordt ook de juiste printer vastgesteld; er is voldoende keus. Het gebruik daarvan is echter iets anders. Een minpunt blijkt het afdrukken van een SWP bestand. In theorie zal het wel kloppen, maar bij het testen kwam vaak zoiets als: 'DVI bestand niet aangemaakt' (terwijl het 'previewen' zonder problemen verliep). Het lijkt iets te maken te hebben met de onjuiste overdracht van tussenbestanden. Ook lijkt het erop dat 'Print' in verschillende 'windows' ook verschillend werkt. Mijn HP-Deskjet500 printer bleek slechts maximaal 5 pagina's af te drukken. Een aangeboden groter DVI bestand (door emTeX gemaakt) liep, na 'alles printen' al in de opbouwfase van de drukopdracht vast bij pagina 7. Het installatieprogramma vertelt dat 16MB swapfile (van Windows) nodig is. Echter: 'mijn Windows' wilde niet meer dan 15,7MB gebruiken, en ik weet niet hoe dat op te lossen.

Een ander probleem is het instellen van de bladspiegel. Het lukte me niet, ook niet na het raadplegen van de documentatie, om die in te stellen (onhandigheid met Windows?). Ter illustratie van de ondervonden problemen: Een keer kwam ik er achter, via de documentatie, dat in een hulpscherm een optie ('reverse order') niet was te vinden omdat dat 'window' voor een deel buiten beeld was gekomen! Zoiets is voor de leek, die ik ben, natuurlijk erg frusterend.

7 Conclusie

Voor mensen die Windows (of WP) georiënteerd zijn en over een krachtige PC met veel geheugen ruimte beschikken is SCIENTIFIC WORKPLACE een krachtige tekstverwerker met een prachtig beeldscherm. Wie bovendien veel wiskunde te documenteren heeft kan met Scientific WordPlace gemakkelijk mooie documenten maken. Er blijven wensen: nederlandse ondersteuning, onbeperkt aantal pagina's afdrukken, gemakkelijker verwerken van TEX commando's via de toetsen bij de 'editor'.

Kleurgebruik in TABLE

J. Hagen

PRAGMA, Onderwijskundig Bureau voor Advies- en Ontwikkelwerk,
Postbus 125, 800 AC Zwolle, (038) 229775

Abstract

Dit artikel beschrijft enkele macro's die het mogelijk maken binnen TABLE cellen van een tabel van een achtergrondkleur of raster te voorzien. We beperken ons tot een wat technische beschrijving en gaan voorbij aan de esthetische kant van de zaak.

Binnen het bij pragma ontwikkelde CONTEXT maken we gebruik van TABLE. Hoewel dit macropakket in principe af en compleet is, is toch besloten om een, zij het beperkte, *shell* rond dit pakket te schrijven. De belangrijkste reden hiervoor lag in de wens en noodzaak een consequente spatiëring bij gebruikers af te dwingen. Dit echter wel met behoud van de volledige functionaliteit van TABLE. Een bijkomend voordeel was — en dat bleek pas veel later — dat kleurgebruik in tabellen vrij eenvoudig te realiseren was.

We beginnen met een voorbeeld van een eenvoudige tabel.

```
\starttabel[|c|c|]
\HL
\VL alfa \VL beta \VL\SR
\HL
\VL alfa \VL beta \VL\FR
\VL alfa \VL beta \VL\MR
\VL alfa \VL beta \VL\LR
\HL
\stoptabel
```

In het bovenstaande voorbeeld staan `\HL` en `\VL` voor een Horizontal Line en een Vertical Line. De overige commando's staan voor: Seperate Row, First Row, Mid Row en Last Row. De aldus gedefinieerde tabel ziet er als volgt uit:

alfa	beta
alfa	beta
alfa	beta
alfa	beta

Geheel in de lijn van de andere commando's binnen CONTEXT, wordt een tabel omringd door `\start-\stop`-commando's. Tussen `[ ]` wordt het kolom-formaat meegegeven. Omdat hierin vierkante haakjes kunnen voorkomen zijn de volgende varianten ook mogelijk.

```
\starttabel[|c|c|]
\starttabel[{|c|c|}]
\starttabel{|c|c|}
```

De genoemde commando's zijn in principe gelijkwaardig aan de door TABLE actief gemaakte karakters `|` en `"`, het spatiëringscommando `\\` en het lijncommando `.` De bovenstaande tabel komt dan ook (in grote lijnen) overeen met:

```
\BeginTable
\BeginFormat | c | c | \EndFormat
\_
| alfa | beta | \\+22
\_
| alfa | beta | \\+20
| alfa | beta | \\+00
| alfa | beta | \\+02
\_
\EndTable
```

Afgezien van de `\Begin..` en `\End..` commando's kunnen alle TABLE-commando's gewoon gebruikt worden, bijvoorbeeld:

```
\starttabel[|c|c|]
\_
| alfa | beta | \SR
\_
| alfa | beta | \FR
| alfa | beta | \MR
| alfa | beta | \LR
\_
\stoptabel
```

Als in de specificatie instellingen als `s0` worden meegegeven moet men zonodig een ruimere layout gebruiken als hierboven, bijvoorbeeld:

```
\starttabel[s0 |c|c|]
```

Een of meer kenmerken van tabellen kunnen worden ingesteld met:

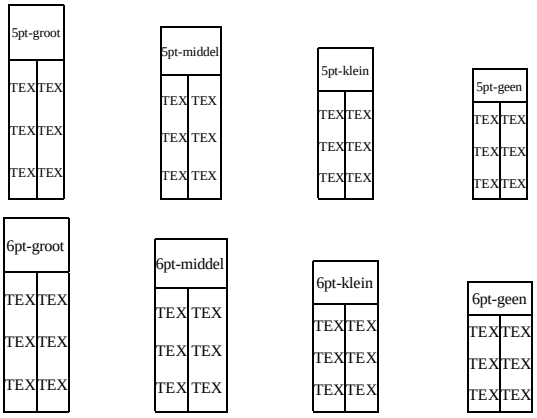
```
\steltabellenin
  [afstand=,
   korps=,
   commandos=]
```

Met behulp van `commandos` kunnen bepaalde instellingen van TABLE zelf worden meegegeven, bijvoorbeeld:

```
\steltabellenin[commandos=\Expand]
```

Dit geeft een tabel met een breedte `\hsize`.

De instellingen `afstand` en `korps` hebben betrekking op de eerder genoemde commando's `\SR`, `\FR`, `\M` en `\LR`.



Er zijn naast de reeds genoemde nog enkele commando's beschikbaar:

```
\starttabel[|c|c|c|]
\HL
\VL alfa  \NC beta  \NC gamma \VL\SR
\DC       \DL       \DC          \DR
\VL beta  \VL gamma \VL alfa  \VL\SR
\DC       \DL       \DC          \DR
\VL gamma \NC alfa  \NC beta  \VL\SR
\HL
\stoptabel
```

Hierin staat `\DL` voor Division Line, `\DC` voor Division Column en `\DR` voor Division Row, terwijl met `\NC` naar de Next Column wordt gesprongen. Naast het laatstgenoemde commando kennen we nog `\NR` wat staat voor Next Row. Deze tabel ziet er als volgt uit:

alfa	beta	gamma
beta	gamma	alfa
gamma	alfa	beta

Het commando `\DC` betreft standaard een kolom en komt overeen met TABLE's commando `\=`. Willen we een lijn trekken over meerdere kolommen, dan kunnen we dat opgeven op de hieronder getoonde wijze.

```
\starttabel[|c|c|c|]
\HL
\VL \LOW{low} \VL \TWO{n/m} \VL\SR
\DC           \DL[2]           \DR
\VL           \VL n \VL m   \VL\SR
\HL
\VL alfa      \VL 1 \VL a   \VL\FR
\VL alfa      \VL 2 \VL b   \VL\MR
\VL alfa      \VL 3 \VL c   \VL\LR
\HL
\stoptabel
```

In plaats van `\TWO` hadden we ook het TABLE-commando `\use2` kunnen gebruiken. In verband met de spatiëring kunnen we echter `\LOW` niet zonder meer vervangen door `\lower`. Deze tabel ziet er als volgt uit:

low	n/m	
	n	m
alfa	1	a
alfa	2	b
alfa	3	c

Samenvattend hebben we dus de volgende commando's voor spatiëring:

commando	**betekenis**
`\SR`	Seperate Row
`\FR`	First Row
`\MR`	Mid Row
`\LR`	Last Row

en de volgende commando's voor lijnen:

commando	**betekenis**
`\HL`	Horizontal Line
`\VL`	Vertical Line
`\DL[getal]`	Division Line
`\DC`	Division Column
`\DR`	Division Row

verder zagen we nog `\LOW` om tekst te verlagen en korte commando's als `\TWO`, `\THREE` enz. om over meerdere kolommen te zetten.

Het feit dat op een dergelijke manier gedefinieerde commando's een consistente spatiëring leveren, bleek het kleuren van tabellen aanzienlijk te vereenvoudigen.

Hoewel de achterliggende macro's wellicht wat ingewikkeld ogen, is het mechanisme verrassend eenvoudig:

1. trek een dikke horizontale lijn
2. spring vertikaal terug
3. plaats de tekstregel

De hoogte en diepte van de lijn dienen gelijk te zijn aan die van de tekstregel, die, zoals we zagen, nauwkeurig vastligt. We hoeven ons (gelukkig) niet te bekommeren om de breedte van de regel, omdat TEX dat zelf uitzoekt. De benodigde commando's vallen daarbij terug op `\noalign`. Uit het feit dat het beschreven mechanisme is te implementeren binnen TABLE benadrukt nog eens hoe goed dit macro-pakket in elkaar zit.

De commando's om achtergronden in grijs of kleur te plaatsen lijken op de eerder beschreven commando's om Division Lines te tekenen.

commando	**betekenis**
`\BL[n,type,spec]`	Background Line
`\BC`	Background Column
`\BR`	Background Row

aangevuld met twee commando's die over de volle breedte van de tabel werken:

commando	**betekenis**
`\CL[specificatie]`	Color Line
`\RL[specificatie]`	Raster Line

In de onderstaande voorbeelden is het gebruik van deze commando's getoond. De regels waarin de achtergronden worden gespecificeerd, gaan *vooraf* aan de regels met tekst. Let op: net als bij `\DL` geldt ook hier dat het commando automatisch naar de volgende kolom gaat. Gebruik dus niet meer `\BC`'s als nodig.

Met `\BR` roept men een laatste specificatie op. Dit commando wordt gevolgd door commando's als `\SR` en `\FR`, die de hoogte van de (nog volgende) tekstlijn aangeven.

Mogelijke achtergronden zijn `color` en `raster`. Als men de achtergrond specificeert, dan kan naast het type een raster of kleur worden opgegeven. Als geen kolom wordt opgegeven wordt uitgegaan van 1 kolom. Eerst wat eenvoudige voorbeelden.

test	test
test	test
test	test
test	test

```
\starttabel[|c|c|]
\BC      \BL          \SR
\HL
\VL test \VL test \VL\SR
\HL
\VL test \VL test \VL\FR
\VL test \VL test \VL\MR
\VL test \VL test \VL\LR
\HL
\stoptabel
```

Met `\BC` gaan we naar kolom 1. Met `\BL` gaan we naar kolom twee, waar we een achtergrond laten aanbrengen.Tot slot delen we met `\SR` mee dat de achtergrondlijn betrekking heeft op een (nog volgende) Seperate Row. Daarbij wordt ook de ruimte tussen de kolommen meegenomen.

De omgekeerde variant ziet er als volgt uit. Bedenk steeds dat met `\BL` eerst naar een volgende kolom springt.

test	test
test	test
test	test
test	test

```
\starttabel[|c|c|]
\BL                  \SR
\HL
\VL test \VL test \VL\SR
\HL
\VL test \VL test \VL\FR
\VL test \VL test \VL\MR
\VL test \VL test \VL\LR
\HL
\stoptabel
```

Twee of meer cellen naast elkaar van een achtergrond voorzien gaat als volgt:

test	test
test	test
test	test
test	test

```
\starttabel[|c|c|]
\BL[2]               \SR
\HL
\VL test \VL test \VL\SR
\HL
\VL test \VL test \VL\FR
\VL test \VL test \VL\MR
\VL test \VL test \VL\LR
\HL
\stoptabel
```

Bij wat bredere tabellen voorkomt een nette setup al snel heel wat problemen.

test	test	test
test	test	test
test	test	test
test	test	test

```
\starttabel[|c|c|c|]
\BL[3]                          \SR
\HL
\VL test \VL test \VL test \VL\SR
\HL
\VL test \VL test \VL test \VL\FR
\VL test \VL test \VL test \VL\MR
\VL test \VL test \VL test \VL\LR
\HL
\stoptabel
```

test	test	test
test	test	test
test	test	test
test	test	test

```
\starttabel[|c|c|c|]
\BC      \BL[2]              \SR
\HL
\VL test \VL test \VL test \VL\SR
\HL
\VL test \VL test \VL test \VL\FR
\VL test \VL test \VL test \VL\MR
\VL test \VL test \VL test \VL\LR
\HL
\stoptabel
```

test	test	test
test	test	test
test	test	test
test	test	test

```
\starttabel[|c|c|c|]
\BC      \BC      \BL          \SR
\HL
\VL test \VL test \VL test \VL\SR
\HL
\VL test \VL test \VL test \VL\FR
\VL test \VL test \VL test \VL\MR
\VL test \VL test \VL test \VL\LR
\HL
\stoptabel
```

test	test	test
test	test	test
test	test	test
test	test	test

```
\starttabel[|c|c|c|]
\BC      \BL                  \SR
\HL
\VL test \VL test \VL test \VL\SR
\HL
\VL test \VL test \VL test \VL\FR
\VL test \VL test \VL test \VL\MR
\VL test \VL test \VL test \VL\LR
\HL
\stoptabel
```

In het onderstaande voorbeeld lijkt een `\BC` te mankeren. Bedenk ook hier dat met `\BL` al naar de volgende kolom wordt gegaan.

test	test	test
test	test	test
test	test	test
test	test	test

```
\starttabel[|c|c|c|]
\BL              \BL          \SR
\HL
\VL test \VL test \VL test \VL\SR
\HL
\VL test \VL test \VL test \VL\FR
\VL test \VL test \VL test \VL\MR
\VL test \VL test \VL test \VL\LR
\HL
\stoptabel
```

Omdat `\SR` een regel afsluit hoeven in de resterende kolommen geen `\BC`'s te worden opgenomen.

test	test	test
test	test	test
test	test	test
test	test	test

```
\starttabel[|c|c|c|]
\BC      \BL                  \SR
\HL
\VL test \VL test \VL test \VL\SR
\HL
                             \BR\FR
\VL test \VL test \VL test \VL\FR
                             \BR\MR
\VL test \VL test \VL test \VL\MR
                             \BR\LR
\VL test \VL test \VL test \VL\LR
\HL
\stoptabel
```

Tot slot wat voorbeelden van nog bredere tabellen. We zien dat in de laatste twee voorbeelden slechts weinig commando's nodig zijn om een rij te kleuren.

aa	bb	cc	dd
aa	bb	cc	dd
aa	bb	cc	dd
aa	bb	cc	dd

```
\starttabel[|c|c|c|c|]
\BC    \BL[r,0.7]    \BL[r,0.9] \SR
\HL
\VL aa \VL bb \VL cc \VL dd  \VL\SR
\HL
                             \BR\FR
\VL aa \VL bb \VL cc \VL dd  \VL\FR
                             \BR\MR
\VL aa \VL bb \VL cc \VL dd  \VL\MR
                             \BR\LR
\VL aa \VL bb \VL cc \VL dd  \VL\LR
\HL
\stoptabel
```

Met `\BR` roepen we de meest recente specificatie op. We hoeven dus niet opnieuw aan te geven welke kolommen van een achtergrond worden voorzien. Wel moet de spatiëring worden opgegeven.

De eerste regel in het bovenstaande voorbeeld had ook korter gekund, namelijk:

```
\BC \RL[0.7] \RL[0.9] \SR
```

In een tabel kunnen zowel rasters als kleuren worden gebruikt. Rasters kunnen ook in termen van kleur kunnen worden gedefinieerd. Een raster met een gradatie (grijswaarde) $Gr = .9$ komt overeen met een kleur met de RGB-waarden $r = g = b = .9$. Vaak kan men dan ook volstaan met alleen kleuren, waaronder enkele grijswaarden.

aa	bb	cc	dd
aa	bb	cc	dd
aa	bb	cc	dd
aa	bb	cc	dd

```
\starttabel[|c|c|c|c|]
\BC    \BL[r,0.8]    \BL[c,rood] \SR
\HL
\VL aa \VL bb \VL cc \VL dd   \VL\SR
\HL
                              \BR\FR
\VL aa \VL bb \VL cc \VL dd   \VL\FR
                              \BR\MR
\VL aa \VL bb \VL cc \VL dd   \VL\MR
                              \BR\LR
\VL aa \VL bb \VL cc \VL dd   \VL\LR
\HL
\stoptabel
```

We zien dat bij `\BL` een nadere aanduiding van de achtergrond `r` of `c` wordt meegegeven. Het was ook goed gegaan met `raster` of `color`. Kleuren en rasters mogen door elkaar worden gebruikt.

Het is mogelijk een hele rij van een achtergrond te voorzien met `\RL` en `\CL`, zonder nummers mee te geven.

aa	bb	cc	dd
aa	bb	cc	dd
aa	bb	cc	dd
aa	bb	cc	dd

```
\starttabel[|c|c|c|c|]
\RL\FR
\VL aa \VL bb \VL cc \VL dd \VL\FR
\VL aa \VL bb \VL cc \VL dd \VL\MR
\RL\MR
\VL aa \VL bb \VL cc \VL dd \VL\MR
\VL aa \VL bb \VL cc \VL dd \VL\LR
\stoptabel
```

aa	bb	cc	dd
aa	bb	cc	dd
aa	bb	cc	dd
aa	bb	cc	dd

```
\starttabel[|c|c|c|c|]
\CL[groen]\SR
\VL aa \VL bb \VL cc \VL dd \VL\SR
\VL aa \VL bb \VL cc \VL dd \VL\FR
\VL aa \VL bb \VL cc \VL dd \VL\MR
\VL aa \VL bb \VL cc \VL dd \VL\LR
\stoptabel
```

De volgende (aanroepen van) commando's zijn equivalent:

```
\BL[c,...]  \BL[color,...]   \COLOR[...]
\BL[r,...]  \BL[raster,...]  \RASTER[...]
```

De oplettende lezer zal zijn opgevallen dat ook als geen specificatie is meegegeven cellen van een achtergrond worden voorzien. Deze standaardachtergronden zijn in te stellen met:

```
\steltabellenin
  [achtergrondkleur=,
   achtergrondraster=,
   achtergrond=]
```

Hierbij kan `achtergrond` de waarde `kleur` of `raster` krijgen. De defaultwaarde is `raster`. Als `achtergrondkleur` kan de naam van een kleur worden meegegeven en als `achtergrondraster` een getal tussen 0 en 1. Achtergronden zijn echter alleen zichtbaar als de betreffende koppelingen zijn gelegd met het omhullende macro-pakket.

Natuurlijk zijn de hier beschreven commando's voor uitbreiding en verbetering vatbaar. Omdat we in de praktijk nog geen problemen zijn tegengekomen die onoplosbaar zijn, zien we hiervan voorlopig af. De ontwikkelingen volgen de vraag.

Helaas blijkt het mechanisme dat `\.rules` plaatst niet altijd even nauwkeurig te werken. Of de oorzaak hiervan bij mij, bij TEX of bij de DVI-drivers ligt is mij nog onduidelijk.

Het is belangrijk dat de Horizontal Rules (`\HL`) worden geplaatst *nadat* de achtergrond is geplaatst. Anders wordt de achtergrond voorgrond en verdwijnt (een deel van) de lijn. De eerdere voorbeelden illustreren hoe het moet; het onderstaande voorbeeld geeft aan hoe het *niet* moet.

aa	bb	cc	dd
aa	bb	cc	dd
aa	bb	cc	dd
aa	bb	cc	dd

```
\starttabel[|c|c|c|c|]
\BC    \BL[c,groen]  \BL[c,rood] \SR
\HL
\VL aa \VL bb \VL cc \VL dd   \VL\SR
                              \BR\FR
\HL
\VL aa \VL bb \VL cc \VL dd   \VL\FR
                              \BR\MR
\VL aa \VL bb \VL cc \VL dd   \VL\MR
                              \BR\LR
\VL aa \VL bb \VL cc \VL dd   \VL\LR
\HL
\stoptabel
```

De getoonde voorbeelden zien we nergens twee gekleurde kolommen naast elkaar. Dit kan namelijk niet, althans, niet zonder ingewikkelde constructies. Een mogelijke oplossing is het introduceren van dummy-kolommen:

aa	bb	cc
aa	bb	cc
aa	bb	cc

```
\starttabel[|c||c||c|]
\BL[c,groen] \BL[c,rood]         \FR
\NC aa    \NC\NC bb \NC\NC cc \NC\FR
                              \BR\MR
\NC aa    \NC\NC bb \NC\NC cc \NC\MR
                              \BR\LR
\NC aa    \NC\NC bb \NC\NC cc \NC\LR
\stoptabel
```

We zien dat de afstand tussen de kolommen wat groter is dan wellicht gewenst. Dit is op te lossen door de TABLE-variabelen `\InterColumn...` aan te passen. Van een

alternatieve aanpak, namelijk achter de schermen gebruik maken van in plaats van \= is afgezien, omdat dit minder mooie tabellen oplevert.

Experimenteren staat vrij. De onderstaande tabel toont dat we kunnen smokkelen met de spatiëring. In dit geval moeten wel alle tekstregels worden afgesloten met een \SR.

aa	bb	cc
aa	bb	cc
aa	bb	cc

```
\starttabel[|c||c||c|]
\BL[c,groen]        \BL[c,rood] \MR
\NC aa       \NC bb \NC cc   \NC\SR
                             \BR\MR
\NC aa       \NC bb \NC cc   \NC\SR
                             \BR\MR
\NC aa       \NC bb \NC cc   \NC\SR
\stoptabel
```

De macro's zijn afgestemd op het boven op Plain TEX geschreven CONTEXT. Ze blijken echter ook goed te werken met LATEX, mits eerst enkele ondersteunende macro's worden geladen. Bovendien zullen enkele koppelingen met LATEX moeten worden gelegd. De volgende files worden gebruikt:

- `m-tabel.tex` : de eigenlijke macro's
- `m-tabel.sty` : de koppeling met pakketten
- `m-table.sty` : de engelstalige interface
- `m-cont-a.tex` : ondersteunende macro's
- `m-cont-s.tex` : enkele surrogaat macro's

In de file `m-cont-s` worden eventuele koppelingen met andere macropakketten gerealiseerd. De in `m-cont-a` opgenomen ondersteunende macro's zijn voor meerdere doeleinden bruikbaar. Ze ondersteunen onder andere de meertalige interface binnen CONTEXT. Binnen pakketten als LATEX kan worden volstaan met een vermelding in de stijldefinitie, bijvoorbeeld:

```
\documentstyle[...,m-tabel,...]{...}
```

Kleur moet natuurlijk wel ondersteund worden. De engelstalige interface wordt geladen met door in de bovenstaande specificatie `m-tabel` te vervangen door `m-table`.

TEX-verwerking in kleur

J. Hagen

PRAGMA, Onderwijskundig Bureau voor Advies- en Ontwikkelwerk,
Postbus 125, 800 AC Zwolle, (038) 229775

1 Syntax-controle

Vooral beginnende TEX-gebruikers ervaren de mix van commando's en tekst als verwarrend. Mede om die reden hebben wij zo'n vier jaar geleden onze tekstverwerker afgestemd op het gebruik van TEX, door TEX-commando's op een wat afwijkende manier weer te geven. Het blijkt dat, afgezien van aanvullende mogelijkheden om de syntax te controleren, het aantal fouten in commando's drastisch afneemt. We laten hier een stukje tekst zien.

```
\starttekst

\hoofdstuk [ hfd:intro ] { Inleiding }

We gaan het hebben over:

\startopsomming [ n,opelkaar ]
\som    alfa
\som    beta  ( $ \beta$ )
\som    gamma
\stopopsomming

Succes ermee.

\stoptekst
```

Het gekleurd weergeven van macro's en TEX-specifieke karakters maakt ogenschijnlijk lastige macro's — bijvoorbeeld in macro-pakketten of style-files — vaak veel inzichtelijker. Het gebruik van kleur voor TEX-specifieke toepassingen was mede mogelijk omdat we tot nu toe geen gebruik hadden gemaakt van kleurgebruik in onze programmatuur.

Commando's worden in groen weergegeven. Daarbij wordt rekening gehouden met vaak gebruikte speciale karakters, zoals: @, !, ? en *, bijvoorbeeld:

```
 \commando
\?commando
\!!commando
\commando*
\c@mm@nd@
```

Rood is gereserveerd voor argumenten en de overgang maar de wiskundige mode:

```
tekst  { tekst }  tekst
tekst  $ formule $  tekst
```

Cyaan, dat beter voldoet dan blauw, gebruiken we voor haakjes:

```
tekst [ziezo] tekst (ziezo) tekst
```

Bovendien gebruiken we blauw voor <, > en =. Dit enerzijds om formules beter leesbaar te maken, anderzijds worden deze karakters gebruikt in definities van macro's. Om die reden is ook het parameter-karakter # cyaan. Hoewel niet vaak gebruikt, wordt " eveneens cyaan gekleurd. Dit karakter heeft in TABLE een speciale functie.

Een vierde gebruikte kleur is geel. Deze kleur is gereserveerd voor +, - en /. Dit komt zowel van pas in formules als in samengestelde woorden. Resten nog enkele vaak gebruikte karakters als %, ~, ‘, ’, |, &, ^ en _.

2 Spellingcontrole

Een tweede voorbeeld van kleurgebruik bij TEX-verwerking betreft spellingcontrole. In plaats van het (vaak vermoeiende) vraag-antwoord-spelletje, kan real-time de spelling van woorden worden gecontroleerd. Daarbij blijven de TEX-commando's natuurlijk buiten beschouwing. Goed geschreven woorden worden groen weergegeven, onbekende woorden rood. Woorden van vier karakters of minder, worden niet gecontroleerd, eenvoudigweg omdat veel fouten bestaande woorden zijn.

```
Dit stukje tekst bevat weinig fouten.

Alleen  dit wooord  is verkeerd ge-
schreven.
```

Als we al controlerend door de tekst lopen, kunnen we de rode woorden verbeteren. Deze worden dan groen. Trema's en dergelijke worden automatisch geplaatst en hoeven niet te worden verbeterd.

Om een indruk te krijgen van de schrijfstijl worden vervoegingen van worden, moeten, zullen en dienen geel weergegeven, bijvoorbeeld:

```
Men dient zich er terdege van bewust
te zijn dat er nauwlettend toezicht
wordt gehouden!
```

Andere vormen van controle op schrijfstijl zijn ook mogelijk, bijvoorbeeld op het gebruik van woorden als die, dat en deze of aanspreekvormen als wij, ik of jij. Dergelijke controles zijn soms handig in studieteksten.

De hier beschreven manier van spellingcontrole is indertijd geïmplementeerd onder druk van gebruikers die niet meer zonder een vorm van controle konden. Voorbijgaand aan de discussie of spellingcontrole nu werkelijk tot betere teksten leidt, kunnen we toch voorzichtig de volgende conclusies trekken:

- Fouten zitten meestal in wat langere woorden. Korte woorden worden minder vaak fout geschreven maar komen soms wel eens dubbel voor. Omdat je onbewust vooral let op de niet-groene (dus goed geschreven) woorden, haal je fouten in (gebruik van) kleine woorden er meestal wel uit.
- Foute (of onbekende) woorden worden in rood weergegeven en kunnen direkt worden verbeterd. Niet zelden blijkt dat onbekende woorden beter kunnen worden vervangen door hun eerder gebruikte synoniem, iets wat in opleidingsmateriaal geen kwaad kan.
- Voorbijgaand aan een discussie over schrijfstijlen, worden teksten er soms beter op als al te veel passieve werkwoorden worden vermeden. Waarschuwen werkt in dit geval beter dat verbeteren, vandaar de zacht-gele kleur.
- Van de mogelijkheid aanspreekvormen en dergelijke te controleren wordt zelden gebruik gemaakt.
- Tijdens spellingcontrole heeft men nauwelijks in de gaten dat er TEX-commando's in de tekst voorkomen.

3 De programma's

Beide beschreven controles in kleur zijn beschikbaar in het programma `texedit`, dat naast deze controles ook wat meer geïntegreerde manier van TEX-verwerken ondersteunt. De uitgeklede versie van dit programma (`wdt`) biedt, naast natuurlijk de standaard tekstverwerkingshandelingen, alleen een controle van TEX-commando's. Beide programma's hebben als kern dezelfde tekstverwerker. Kenmerkend aan deze tekstverwerker, die we inmiddels zo'n acht jaar in onze organisatie gebruiken, is dat de meeste handelingen onder slechts één toets beschikbaar zijn. Bovendien kunnen tegelijk meerdere zeer grote teksten worden geladen (megabytes).

Geïnteresseerden kunnen een kopie van `wdt` bij ondergetekende aanvragen. Het andere programma zal alleen bij voldoende belangstelling in het public domain worden gebracht.

Een zwart-wit kijk op kleur

J. Hagen & J. Jonker

PRAGMA, Onderwijskundig Bureau voor Advies- en Ontwikkelwerk,
Postbus 125, 800 AC Zwolle, (038) 229775

Abstract

Wie tegenwoordig op een zwart-wit tv-toestel een uitzending volgt zal zich vaak tevreden moeten stellen met een weinig contrastrijk beeld. Na de introductie van de kleuren-tv is nog lang rekening gehouden met zwart-witkijkers. De keus van kleuren in decors werd mede afgestemd op de weergave in zwart-wit. Dit is niet verwonderlijk, omdat de zwart-witkijkers een ruime meerderheid vormden.
Op papier lijkt zich een dergelijke ontwikkeling af te spelen. Voor kleurenprinters geschikte illustraties, kunnen op zwart-wit printers een matig beeld opleveren. Het is dan ook de vraag hoe we aan de 'wensen' van de overgrote meerderheid van zwart-wit printers tegemoet kunnen komen.

1 Inleiding

Teksten die voor opleidingsdoeleinden worden gebruikt zijn vaak rijkelijk voorzien van illustraties. Met het toenemen van de mogelijkheden van computers en de programma's die daarop draaien, nemen ook de vaak nauwelijks begrensde wensen van de gebruikers toe. De ontwikkelaar van opleidingsmateriaal grijpt de kans om zijn materiaal aantrekkelijk te maken en de grafisch vormgever verkent en verlegt de grenzen van zijn creativiteit.

De kans dat in deze, op zich stimulerende, situatie een probleem ontstaat is niet denkbeeldig. Teksten worden meestal in grijswaarden (zwart-wit) geprint of gedrukt. Illustraties die in het opleidingsmateriaal zijn opgenomen, worden daarnaast vaak ook op transparanten gezet, die in toenemende mate in kleur worden afgedrukt. Bij het afdrukken van een illustratie in grijswaarden (op papier) en in kleur (op transparant) dient het probleem zich aan: de illustraties die er in de tekenpakketten in kleur fraai uitzien, leveren op papier een grauw beeld op. Dit is een gevolg van de vertaalslag van kleur naar grijswaarden. In figuur 1 zijn twee varianten van dezelfde figuur naast elkaar weergegeven. In beide varianten is gebruik gemaakt van kleur. Waar het resultaat in kleurendruk er prima uitziet, levert de links geplaatste variant in zwart-witdruk een egaal grijs gekleurde figuur op. De vraag ligt dan ook voor de hand hoe we een figuur geschikt kunnen maken voor weergave in zowel grijswaarden als kleur.

In dit artikel reiken we een oplossing aan voor het geschetste probleem. We gaan in op de achtergronden van het omzetten van kleuren in grijswaarden en presenteren een systeem waarmee weergave in zowel kleur als zwart-wit mogelijk is. We laten ons daarbij leiden door zowel praktische als esthetische motieven.

2 Kleurgebruik

Voordat we ingaan op kleurgroepen en kleurpaletten staan we even stil bij kleurgebruik. Tufte (1990) onderscheidt vier functies van kleur:

1. to label (color as noun)
2. to measure (color as quantity)
3. to represent or initiate reality (color as representation)
4. to enliven or decorate (color as beauty)

De laatste functie spreekt voor zich. De eerste drie functies illustreert hij aan de hand van een landkaart: (1) water en land, ijsvlaktes en weidegrond worden door kenmerkende kleuren gescheiden, (2) hoogteverschillen worden door nuances in kleur weergegeven en (3) blauwe lijnen worden als vanzelfsprekend voor rivieren aangezien.

In Bolder et al. (1990) wordt een indeling gemaakt in typografische en voor de leesbaarheid belangrijke functies:

1. psychologisch: invloed op stemming en gevoel
2. esthetisch: als toegevoegde waarde
3. accentuerend: om zaken te benadrukken
4. identificerend: om zaken te onderscheiden

Kijkend naar beide typeringen van kleurgebruik, zal duidelijk zijn dat een verkeerde keuze voor kleur grote gevolgen kan hebben voor niet alleen de typografische kwaliteit van een tekst, maar ook voor de effectiviteit.

Tufte geeft een aantal concrete aanwijzingen voor het selecteren van kleuren:

> *Use colors found in nature, especially those on the lighter side such as blues, yellows and grays of sky and shadow.*

Hij toont enige voorbeelden van in zijn ogen verantwoord kleurgebruik en laat tevens zien hoe het niet moet: colorjunk. We zullen zien dat aan zijn wens om lichte kleuren te gebruiken niet eenvoudig kan worden voldaan, omdat deze vrijwel dezelfde grijswaarden opleveren.

Als we op zoek gaan naar de ‘juiste’ kleuren kunnen we niet voorbijgaan aan grijs als kleur:

> *Color spots against a light gray or muted field highlight and italicize data and also help to weave an overall harmony.*

Om aan deze aanbeveling van Tufte tegemoet te kunnen komen, zullen we dus naast kleuren ook grijswaarden beschikbaar moeten hebben.

Als een auteur naar een onderdeel van een illustratie verwijst, zal hij dit doen in zinvolle termen. Hij zal bijvoorbeeld kunnen verwijzen naar de haren op de poten van de spin. Een verwijzing in termen van lichtgeel of donkerrood is weinig zinvol. Ten eerste liggen dergelijke keuzes meestal bij de vormgever, daarnaast is het verschil tussen licht en donker moeilijk eenduidig te definiëren.

Uitgaande van zes te gebruiken kleuren, zou een specificatie van bijvoorbeeld een spuitgietmachine kunnen plaatsvinden in betekenisvolle aanduidingen. Dergelijke aanduidingen zijn altijd positief geformuleerd:

1. belangrijk onderdeel
2. gevaarlijke situatie
3. instelpunt
4. controlepunt
5. dragend deel
6. bewegend deel

Negatieve aanduidingen als *onbelangrijk onderdeel* en *ongevaarlijke situatie* zijn niet nodig, omdat men vanuit de tekst zelden naar iets onbelangrijks verwijst. De ontwikkelaar en ontwerper zouden van te voren afspraken kunnen maken over een beperkt aantal aanduidingen. Bij voertuigen zouden dat kunnen zijn: *dragend deel* en *bewegend deel* en *controlepunt*. Door hiervoor vaste kleuren te gebruiken bevordert men consistentie. De vormgever kan de overige kleuren dan gebruiken naar eigen creatief inzicht. Bij het bepalen van de namen kunnen we ons laten leiden door de eerder genoemde functies.

3 Kleurgroepen

Er zijn ongeveer 50 systemen om kleuren te definiëren en te organiseren. Zo kennen we het RGB-systeem, dat uitgaat van het optellen van *rode*, *groene* en *blauwe* componenten. Dit systeem vinden we terug in beeldschermen. Het in de drukkerswereld gebruikte CMYK-systeem gaat uit van aftrekken van *cyaan*, *magenta* en *geel* (yellow), eventueel aangevuld met *zwart*. Het verschil tussen beide systemen ligt mede in het feit dat beeldschermen licht uitstralen en drukinkten licht weerkaatsen ofwel absorberen. Een derde systeem is het HSV-systeem, dat aansluiting zoekt bij de schilder die kleuren mengt op zijn palet. Dit systeem gaat uit van *tint*, *verzadiging* en *helderheid* (hue, saturation en value/brightness). Welk systeem ook wordt gekozen, vrijwel altijd worden kleuren gedefinieerd in drie componenten.

In grafische programmatuur worden vaak meerdere kleursystemen ondersteund. Kleuren die in het ene systeem zijn gedefinieerd kunnen naar een ander systeem worden vertaald. Kleuren kunnen ook worden omgezet in grijswaarden. De omzetting van RGB naar grijswaarden vindt plaats conform de YIQ-standaard. Deze is ontworpen voor het weergeven van kleur op zwart-wit tv-toestellen.

$$\begin{aligned} \text{Grijs} &= .30 \times \text{rood} \\ &+ .59 \times \text{groen} \\ &+ .11 \times \text{blauw} \end{aligned}$$

In overzicht 1 zijn vier kleuren in hun RGB-componenten gedefinieerd. In de laatste kolom is de grijswaarde gegeven. De lichte kleuren zijn bij het weergeven in kleur uitstekend naast elkaar te gebruiken. Hetzelfde geldt voor de donkere kleuren.

kleur	rood	groen	blauw	grijs
lichtrood	1.00	0.70	0.70	1.000
lichtgeel	0.50	1.00	0.50	0.795
donkerrood	1.00	0.15	0.15	0.405
donkergeel	0.00	0.65	0.00	0.383

Overzicht 1: *Enkele voorbeelden van RGB-waarden.*

We zien echter dat de grijswaarden van de lichte en donkere kleuren onderling weinig verschillen. Als we de kleuren naast elkaar in grijswaarden en kleur afdrukken, dan zien we dat dit in werkelijkheid ook zo is (zie overzicht 2). De vertaalslag die in de (zwart-wit) printer wordt gemaakt is correct, het probleem komt dus voort uit het gegeven dat bepaalde kleuren een zelfde grijswaarde hebben. Overigens worden zowel kleuren als grijswaarden vaak met behulp van rasters gerealiseerd.

Willen we dus figuren zowel in kleur als in grijswaarden weergeven, dan moet het kleurgebruik daarop worden afgestemd. We kunnen alleen die kleuren gebruiken waarvan de grijswaarden verschillen. Als vertrekpunt kan een serie grijswaarden met goed zichtbare verschillen dienen, zoals hierboven.

Helemaal links is een grijswaarde van 0.95 weergegeven, gevolgd door 0.90, 0.80, 0.70 enz. De laatste drie waarden liggen zo dicht tegen zwart aan dat we ze verder buiten beschouwing laten. We houden dus acht bruikbare gradaties over. We noemen (hier) zo’n serie een kleurgroep.

De oplossing van ons kleurprobleem ligt in het vinden van vergelijkbare groepen voor de kleuren rood, groen, blauw, cyaan, magenta en geel.

In overzicht 3 is een rode kleurgroep weergegeven. Bij weergave in zwart-wit zijn onder elkaar 8 grijsgradaties te zien. In kleur is echter de linkerhelft van een balkje rood en de rechterhelft grijs. Rechts zijn de RGB-waarden weergegeven. Enkele groepen lijken op het eerste gezicht voor verbetering vatbaar. In de praktijk blijken de in dit artikel getoonde groepen echter goed te voldoen.

Op analoge wijze kunnen we voor de andere genoemde kleuren passende RGB-waarden vinden. In overzicht 4 zijn de zes kleurgroepen weergegeven.

Deze kleurgroepen zijn grotendeels proefondervindelijk, maar niet willekeurig tot stand gekomen. Het vergelijken van de kleurverlopen op verschillende beeldschermen en grijswaarden op printers ligt hieraan ten grondslag. De eerder gepresenteerde formule is daarbij behulpzaam geweest.

Formule 1 is ook te schrijven als:

$$Gr = 0.30r + 0.59g + 0.11b$$

Hierin staan Gr voor de grijswaarde en r, g en b voor respectievelijk de rode, groene en blauwe component. Het blijkt dat als $r = g = b$, de bijdrage van de verschillende kleurcomponenten aan de grijswaarde nogal verschillen. Deze relaties vertonen overduidelijk overeenkomst met de gevoeligheid van het oog voor de drie kleuren. In de eerder genoemde yig-standaard wordt nadrukkelijk rekening gehouden met de karakteristieken van onze visuele waarneming (Watt, 1990).

$$\text{bijdrage}_b < \text{bijdrage}_r < \text{bijdrage}_g$$

Verandering van de groencomponent heeft dus meer effect op een grijswaarde dan een verandering in blauw.

Formule 1 kan behulpzaam zijn bij het bepalen van vergelijkbare grijswaarden. Stel dat we bij de grijswaarde 0.70 geschikte kleurcomponenten voor cyaan zoeken. Als we $b = 1$ stellen dan geldt:

$$0.70 = 0.30r + 0.59g + 0.11$$

Dit is te vereenvoudigen tot:

$$g = 1.00 - 0.51r$$

Vullen we voor r achtereenvolgens 0.30, 0.40 en 0.50 in, dan vinden we 0.85, 0.80 en 0.75 voor g. Uiteindelijk is gekozen voor de combinatie $r = 0.40$, $g = 0.80$ en $b = 1.00$. In overzicht 5 zijn de drie varianten weergegeven.

De vereenvoudigde formule 5 ziet er bij andere waarden van b en Gr natuurlijk anders uit.

4 Kleurpaletten

Een goede weergave in zowel grijswaarden als in kleur is te realiseren door het aantal kleuren in een groep te beperken tot 8, wat bij het vormgeven van figuren meestal geen probleem is.

In figuur 3 is de spin nogmaals weergegeven. Links is de grijze variant van de gekleurde figuur weergegeven. De gekleurde figuur is dus zowel in zwart-wit als in kleur te gebruiken.

Het op deze (en andere manieren) kleuren van illustraties valt of staat met consistent kleurgebruik. Nu blinken de meeste tekenprogramma's niet uit in het ondersteunen van seriematig (projectmatig) werken. Kleuren komt meestal neer op het toekennen van een kleur aan een object of groep van objecten. (Voorbeelden van objecten zijn: cirkels, lijnen, vlakken en letters). Als bijvoorbeeld in tien tekeningen mannetjes met veiligheidshelmen voorkomen, dan zou het handig zijn als we de helmen konden labelen, bijvoorbeeld als `helm`, en in alle tekeningen tegelijk deze `helmen` van een kleur konden voorzien. In systemen waarin dit mechanisme niet beschikbaar is, zullen we dus van tevoren goed moeten nadenken over de te gebruiken kleuren, omdat aanpassen achteraf veel tijd kost. Gelukkig kunnen we meestal wel paletten samenstellen.

In de overzichten 7 en 8 is een voorbeeld gegeven van een serie naast elkaar te gebruiken kleuren. De kleuren vormen samen een palet.
Natuurlijk kunnen meerdere gradaties van een kleur gebruikt worden, ook hoeven niet alle kleuren te worden gebruikt. Daarbij kan grijs meedoen als kleur. Alles gaat goed, zolang we in iedere (horizontale) rij maar één kleur gebruiken. In ons geval moeten de nummers dus verschillen.

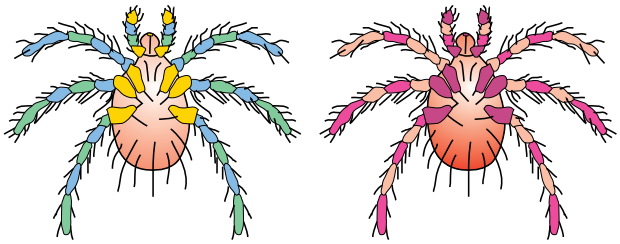
We geven in dit artikel enkele voorbeelden van paletten. Omdat het aantal mogelijke combinaties zeer groot is, moge duidelijk zijn we niet streven naar volledigheid. Met een knipoog naar de natuurkunde gebruiken we zelf de neutrale aanduidingen *top*, *bottom*, *up*, *down*, *charm* en *strange* voor de kleuren. Eventueel kan men een palet nog aanvullen met *low* en *high* voor grijswaarden, bijvoorbeeld voor een lichte achtergrond

5 Opmerking

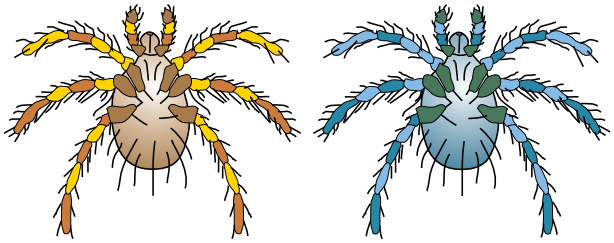
Bij het definiëren van de zes reeds eerder getoonde kleurverlopen hebben rood, groen, blauw, cyaan, magenta en egeel als uitgangspunt gediend. Daarbij deed zich het probleem voor dat (op het beeldscherm) geel snel naar verzadiging neigt en blauw richting paars gaat. Dit is een reden geweest om de gele en blauwe verlopen iets afwijkend te benaderen. Bij het blauwe verloop blijkt dit alleen uit de bijbehorende grijswaarden, bij geel gaan we via oranje naar bruin, wat in kleur goed te zien is. In overzicht 9 zijn de theoretische, met een sterretje gemarkeerde verlopen, naast de acceptabele verlopen gezet.

De eigen ervaring leert dat in kleurendruk cyaan en magenta ook bruikbaar zijn bij het combineren van tekst en kleur. Uit de overzichten 10 en 11 blijkt dat bij het op deze manier combineren van kleuren in grijswaarden het best een vette letter kan worden gebruikt. Afgezien van zwart op geel, rood op geel, wit op blauw, wit op rood en wit op zwart, is kleurgebruik in tekst geen succes.

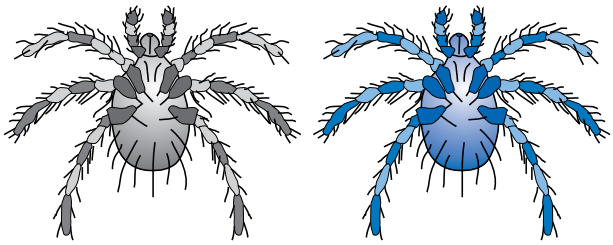
Een extra probleem doet zich voor als teksten niet alleen op papier maar ook interactief worden aangeboden, dat wil zeggen op de computer. Afhankelijk van het voor weergave van de tekst gebruikte programma, kan dan bijvoorbeeld de mogelijkheid worden ingebouwd onderdelen van een figuur aan te klikken. Klikken op een deel van een figuur kan bijvoorbeeld resulteren in het aanbieden van een tekstfragment over het bewuste deel van de figuur. Bij het definiëren van een palet en het maken van afspraken over 'eenduidige' kleuren dient hier ook nog rekening te worden gehouden met het toekennen van kleuren aan die onderdelen waarop geklikt kan worden. Het reeds genoemde mechanisme van het labelen van objecten in een figuur wordt dan haast onmisbaar, zeker als we dezelfde kleur in de tekst willen gebruiken.



Figuur 1 Twee dezelfde figuren in verschillende kleuren.



Figuur 2: *Welke kleuren geven de werkelijkheid het best weer?*



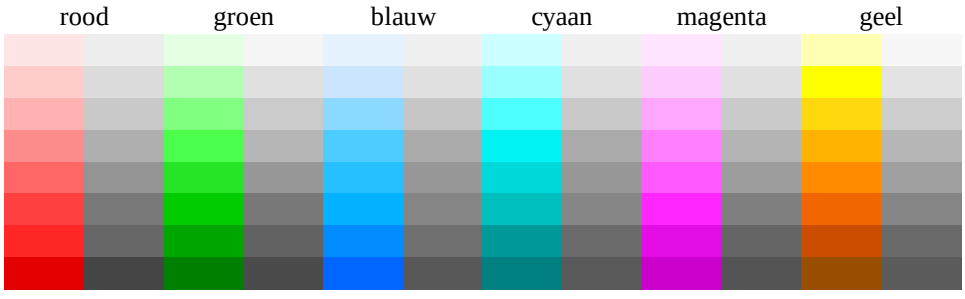
Figuur 3: *Een grijze en een gekleurde spin. Deze spin is zowel in kleur als in zwart-wit goed te zien.*

	grijs	kleur
lichtrood		
lichtgeel		
donkerrood		
donkergeel		

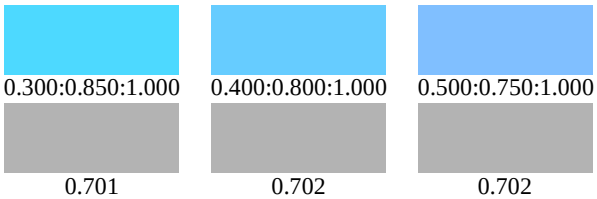
Overzicht 2: *Een vergelijking tussen kleur en grijswaarden van enkele kleuren.*

	rood
1	1.00:0.90:0.90
2	1.00:0.80:0.80
3	1.00:0.70:0.70
4	1.00:0.55:0.55
5	1.00:0.40:0.40
6	1.00:0.25:0.25
7	1.00:0.15:0.15
8	0.90:0.00:0.00

Overzicht 3: *Een vergelijkend overzicht van rode kleuren.*



Overzicht 4: *Zes kleurgroepen met vergelijkbare grijswaarden.*



Overzicht 5: *Drie varianten van cyaan met een zelfde grijswaarde (0.70). De verschillen zijn alleen in kleur te zien.*

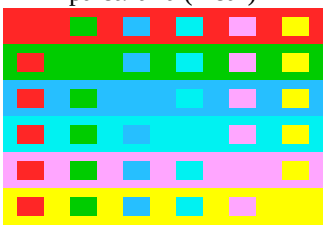
1	2	3	4	5	6	7	8
1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.90
0.90	0.80	0.70	0.55	0.40	0.25	0.15	0.00
0.90	0.80	0.70	0.55	0.40	0.25	0.15	0.00
0.90	0.70	0.50	0.30	0.15	0.00	0.00	0.00
1.00	1.00	1.00	1.00	0.90	0.80	0.65	0.50
0.90	0.70	0.50	0.30	0.15	0.00	0.00	0.00
0.90	0.80	0.55	0.30	0.15	0.00	0.00	0.00
0.95	0.90	0.85	0.80	0.75	0.70	0.55	0.40
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
0.80	0.60	0.30	0.00	0.00	0.00	0.00	0.00
1.00	1.00	1.00	0.95	0.85	0.75	0.60	0.50
1.00	1.00	1.00	0.95	0.85	0.75	0.60	0.50
1.00	1.00	1.00	1.00	1.00	1.00	0.90	0.80
0.90	0.80	0.65	0.50	0.35	0.15	0.05	0.00
1.00	1.00	1.00	1.00	1.00	1.00	0.90	0.80
1.00	1.00	1.00	1.00	1.00	0.95	0.80	0.60
1.00	1.00	0.85	0.70	0.55	0.40	0.30	0.30
0.70	0.00	0.05	0.00	0.00	0.00	0.00	0.00

Overzicht 6: *De RGB-waarden van de kleurgroepen.*

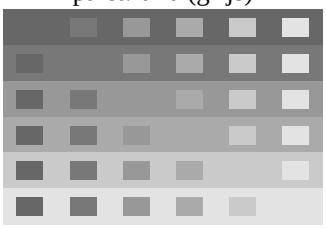
combinatie	%	kleur	grijs
zwart/wit	0.0	tekst **tekst**	tekst **tekst**
groen/wit	−3.0	tekst **tekst**	tekst **tekst**
blauw/wit	−3.4	tekst **tekst**	tekst **tekst**
zwart/geel	−3.8	tekst **tekst**	tekst **tekst**
zwart/rood		tekst **tekst**	tekst **tekst**
rood/geel	−4.8	tekst **tekst**	tekst **tekst**
rood/wit	−8.9	tekst **tekst**	tekst **tekst**
groen/rood	−10.6	tekst **tekst**	tekst **tekst**
wit/blauw		tekst **tekst**	tekst **tekst**
wit/rood		tekst **tekst**	tekst **tekst**
wit/groen		tekst **tekst**	tekst **tekst**
wit/zwart		tekst **tekst**	tekst **tekst**
geel/zwart		tekst **tekst**	tekst **tekst**

Overzicht 10: *Aanbevelingen van Richaudeau, aangevuld met onderzoeksresultaten van Tinker. De getallen drukken het percentage leesbaarheid uit in vergelijking met een zwarte tekst op wit. De tekst is zowel* **vet** *als normaal afgedrukt.*

palet: alfa (kleur)



palet: alfa (grijs)



Overzicht 7: *Een bruikbaar kleurenpalet.*

alfa		
top		1.00:0.15:0.15
bottom		0.00:0.80:0.00
up		0.15:0.75:1.00
down		0.00:0.95:0.95
strange		1.00:0.65:1.00
charm		1.00:1.00:0.00
delta		
top		0.70:0.70:0.00
bottom		0.90:0.90:0.00
up		1.00:1.00:0.00
down		1.00:0.15:1.00
strange		0.30:0.80:1.00
charm		0.90:0.95:1.00

Overzicht 8: *Twee bruikbare paletten.*

Overzicht 9: *Enkele theoretische en bruikbare kleurengroepen naast elkaar.*

combinatie	kleur	grijs
zwart/cyaan	tekst **tekst**	tekst **tekst**
zwart/magenta	tekst **tekst**	tekst **tekst**
cyaan/zwart	tekst **tekst**	tekst **tekst**
magenta/zwart	tekst **tekst**	tekst **tekst**

Overzicht 11: *Enkele aanvullende combinaties van kleur en tekst.*

Hierboven is steeds uitgegaan van een relatief beperkt palet. Van de miljoenen beschikbare kleuren gebruiken we er slechts acht. Hoewel deze bespreking vooral een gevolg is van de randvoorwaarde dat een illustratie er ook in grijswaarden nog goed moet uitzien, is een andere motivatie wellicht van doorslaggevender aard (Watt, 1990):

> *Although colours can be used to highlight logical relationships, flow and so on, they cannot overcome the limitation that human beings can only absorb a certain amount of information at any instant. As the number of available colours (on the display) increases, their distinctiveness is reduced and the complexity of the display increases, counteracting the initial reason for using colours.*

Alle overzichten in dit artikel zijn opgemaakt in een zetsysteem dat het geschetste labelen mogelijk maakt, namelijk TEX. Dit betekent dat bij een verder optimaliseren van het hier getoonde palet, de kleuren in deze figuren en tabellen automatisch worden aangepast. Helaas geldt dit niet voor de figuren. Deze zijn gemaakt in een tekenpalet dat weliswaar zelf gedefinieerde paletten ondersteunt, maar labelen van objecten niet. Wel zijn alle gelijk gekleurde segmenten van de spin aan elkaar gekoppeld, zodat snel een nieuwe kleur kan worden toegekend.

6 Kleur en tekst

Tot slot staan we kort stil bij het gebruik van kleur in combinatie met tekst. Treebus (1991) neemt zijn aanbevelingen voor het in kleur zetten van teksten over van Richaudeau. Daarnaast maakt Rubinstein (1988) melding van onderzoek van Tinker naar de leesbaarheid van tekst in combinatie met kleur.

Als een deel van een tekst in een kleur wordt gedrukt, speelt het probleem van onderscheidbaarheid niet of nauwelijks. Vaak zal men ten hoogste een steunkleur gebruiken, zodat de gekleurde tekst alleen van ‘zwarte’ tekst moet afwijken. Omdat de weergave in grijs met behulp van rasters wordt gerealiseerd, zal het resultaat in zwart-wit zelden bevredigend zijn. Dit stukje tekst is gezet in (donker)groen.

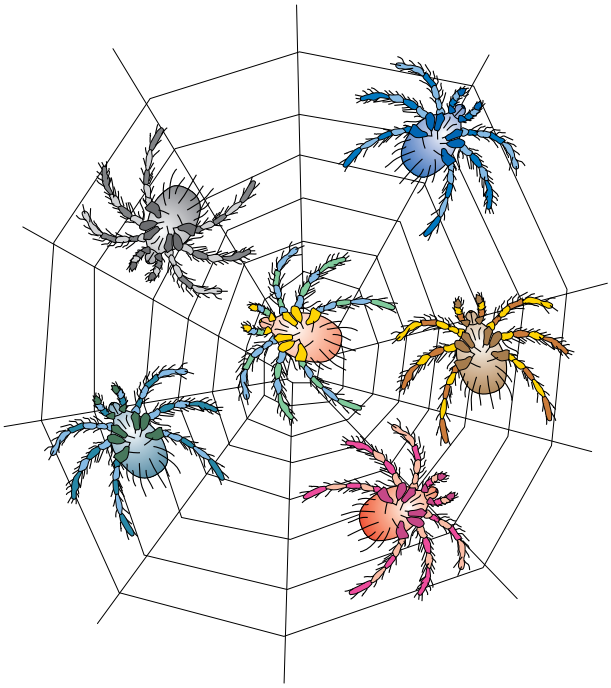
Bij het drukken van een deel van een tekst in kleur moet men wat donkerder kleuren gebruiken.

7 Samenvatting

Samengevat kunnen we concluderen dat illustraties zodanig kunnen worden opgemaakt dat zowel weergave in grijswaarden als in kleur mogelijk is. Ervan uitgaande dat in de grafische programmatuur het specificeren van kleuren, bijvoorbeeld in RGB-waarden, mogelijk is, dient men de volgende stappen te doorlopen:

1. Kies een aantal kleuren.
2. Definieer binnen deze kleuren overeenkomstige grijsverlopen.
3. Zet de zo onstane kleurgroepen tegen elkaar uit in een matrix.
4. Stel uit deze matrix een palet samen van acht kleuren, waarbij elke gradatie maar eenmaal voorkomt.
5. Leg de relatie tussen bepaalde kleuren in een palet en onderdelen van illustraties vast.

De genoemde keuzes dienen de ontwikkelaar en vormgever in overleg te maken. De getallen zijn niet dwingend maar blijken in de praktijk goed te voldoen.



8 Kleurgebruik in TEX

We sluiten dit artikel af met een korte beschrijving van het gebruikte systeem, dat deel uitmaakt van het in TEX geschreven macropakket CONTEXT.

Om een tekst daadwerkelijk in kleur te zetten moet het kleursysteem worden geactiveerd. Dit gebeurt met het commando:

```
\stelkleurenin[status=start]
```

Kleuren worden gedefinieerd met het commando:

```
\definieerkleur[naam][r=,g=,b=]
```

Waarbij aan `r`, `g` en `b` een getal tussen 0 en 1 kan worden toegekend. Als dat beter uitkomt, mogen CMYK-waarden worden opgegeven. In plaats van een specificatie mag tussen de tweede set `[ ]` een naam worden opgegeven van een reeds bestaande kleur.

Definities vinden bij voorkeur plaats in een aparte definitiefile met de naam `colo-xxx.tex`. Standaard is de file `colo-rgb` beschikbaar. Zo’n file wordt geladen met het commando:

```
\stelkleurin[xxx]
```

Een overzicht van de in een definitiefile gedefinieerde kleuren kan worden opgeroepen met:

```
\toonkleur[xxx]
```

Een stuk tekst kan worden gekleurd met behulp van de volgende commando's:

```
\kleur[naam]{tekst}
```

en

```
\startkleur[naam]
... tekst ...
\stopkleur
```

De zwart-wit equivalent van een kleur kan worden opgeroepen met:

```
\grijs[naam]{tekst}
```

Als een naam uniek is, kan ook worden overgegaan op een andere kleur met `\naam`. Het wisselen van kleur is in dat geval te vergelijken met het wisselen van font.

De getalswaarden van kleuren worden opgeroepen met:

```
\kleurwaarde{naam}
```

en de grijswaarde met `\grijswaarde`.

Een kleurgroep wordt gedefinieerd met:

```
\definieerkleurgroep
  [groep][r1:g1:b1,r2:g2:b2:,...]
```

De rode kleurgroep is bijvoorbeeld gedefinieerd met:

```
\definieerkleurgroep
  [rood]
  [1.00:0.90:0.90,
   1.00:0.80:0.80,
   1.00:0.70:0.70,
   1.00:0.55:0.55,
   1.00:0.40:0.40,
   1.00:0.25:0.25,
   1.00:0.15:0.15,
   0.90:0.00:0.00]
```

Een kleur krijgt in een kleurgroep automatisch een volgnummer toegekend. In de lijn van het eerdere betoog kunnen we een kleurgroep controleren met het commando:

```
\toonkleurgroep[groep][opties]
```

Waarbij als opties kunnen worden meegegeven: `horizontaal`, `vertikaal`, `naam`, `waarde` en/of `nummer`. Meerdere instellingen worden gescheiden door een komma. Uit de eerder gegeven voorbeelden van kleurgroepen kan men afleiden waartoe deze opties leiden. Overzicht 3 is bijvoorbeeld opgeroepen met:

```
\toonkleurgroep
  [rood][vertikaal,naam,nummer,waarde]
```

Bij het tonen worden de kleurgroepen deels in grijs weergegeven.

Een ander overzicht kan worden opgeroepen met het commando:

```
\vergelijkkleurgroep[naam]
```

Dit commando laat alle combinaties binnen een kleurgroep zien.

Een palet wordt gedefinieerd met het commando:

```
\definieerpalet
  [naam][label=kleur,label=kleur,...]
```

Een kleur uit een kleurgroep wordt opgeroepen op naam en volgnummer, zoals hieronder is te zien:

```
\definieerpalet
  [alfa]
  [     top=rood:7,
     bottom=groen:6,
         up=blauw:5,
       down=cyaan:4,
    strange=magenta:3,
      charm=geel:2]
```

Net als bij kleurgroepen kunnen ook paletten worden vergeleken:

```
\toonpalet[naam][opties]
\vergelijkpalet[naam]
```

De labels in een palet hoeven niet uniek te zijn. We kunnen dus verschillende paletten met dezelfde labels definiëren. Sterker nog, dit is zelfs aan te bevelen. Er is namelijk steeds slechts een palet actief. Het actieve palet wordt ingesteld met:

```
\stelpaletin[naam]
```

Bij het oproepen van een kleur, bijvoorbeeld met `\kleur`, wordt namelijk eerst gezocht in het actuele palet. Pas als de opgegeven naam niet in het palet voorkomt, wordt gekeken of het op een andere manier is gedefinieerd, bijvoorbeeld met `\definieerkleur`. Het instellen van een palet is dus te vergelijken met het kiezen van een lettertype.

Een complicatie in het gebruik van kleur binnen TEX ligt in het kleuren van tekst die over de paginagrens heen gaat. Er moet namelijk rekening worden gehouden met de nog te plaatsen hoofd- en voetregels en met het feit dat bladzijden onafhankelijk van elkaar moeten worden gereproduceerd. Op iedere pagina moet dus opnieuw de actuele kleur worden ingesteld. Aangezien TEX altijd wat verder is dan de huidige pagina, moet worden bijgehouden wat de actuele kleur is. Hierbij moet rekening worden gehouden met nesting.
Kleurgebruik over pagina's heen levert weinig problemen op, mits een kleine ingreep in de outputroutine plaatsvindt. Voor de meeste toepassingen zijn geen speciale drivers nodig en kan TEX alles zelf afhandelen. Hetzelfde geldt voor achtergrondkleuren. Het valt echter buiten het bestek van dit artikel hierop dieper in te gaan.

9 Literatuur

1. Adobe Systems Incorporated, *Postscript Language Reference Manual.* (1990). Reading, Massachusetts:Addison Wesley Publishing Company.
2. Bolder, T., J. Klinkenberg, H. van Krimpen e.a., *Typografie: uitgangspunten, richtlijnen en techniek.* Amsterdam: Gaade Uitgevers.
3. Rubinstein, R., *Digital Typography: An Introduction to Type and Composition for Computer System Design.* (1988) Reading, Massachusetts: Addison Wesley Publishing Company.
4. Treebus, K.F., *Vormwijzer: Een gids bij het vormgeven en produceren van drukwerk.* (1991) 's-Gravenhage: SDU Uitgeverij.
5. Tufte, E.R., *Envisioning Information.* (1990) Cheshire, Connecticut: Graphics Press.
6. Watt, A.H., *Fundamentals of Three-Dimensional Computer Graphics.* (1990) Reading, Massachusetts: Addison Wesley Publishing Company.

Genezen van WPosis — nu heb ik chronische TEXitis…

Herman Haverkort

herman@fgbbs.iaf.nl

April 1995

Abstract

Bij het vormgeven van een voorstel voor reglementen overschreed een verstokte WordPerfect-gebruiker de grenzen van zijn geliefde tekstverwerker. Frans Goddijn hielp hem aan de TEX. In dit artikel geeft een nieuwe TEX-gebruiker een globale indruk van zijn laatste ervaringen met WordPerfect en zijn eerste ervaringen met TEX. Hij doet dit aan de hand van het werk aan een document waarin het mogelijk moest zijn om vele typografische middelen onafhankelijk van elkaar te gebruiken om verschillende soorten passages aan te duiden. Dit bleek in TEX goed te realiseren, zij het met veel inspanning, en het resultaat was zeer bevredigend. Toen het eerste document eenmaal gereed was, bleek het bovendien daarna *zeer* eenvoudig om soortgelijke documenten te zetten.

Wat vooraf ging

Het was 9 december 1994. Voor die dag stonden de volgende activiteiten op het programma:

1. laat opstaan (te Arnhem);
2. een artikel voor een verenigingsblad afmaken ('s avonds deadline);
3. me scheren (met tegenzin) in verband met:
4. receptie (te Almere);
5. artikel afleveren bij de hoofdredacteur (te Arnhem);

Het eerste punt van het programma wist ik zoals gewoonlijk met groot succes te volbrengen, maar met het tweede had ik aanzienlijk meer moeite.

U moet weten dat ik al een jaar of zes een verstokte WordPerfect-gebruiker was. Met WP kon ik alles voor elkaar krijgen wat ik wilde. Er waren wel eens wat kleine probleempjes, maar ach… daar viel best mee te leven. WP liep wel eens vast (om een indruk te geven: zo vaak dat mijn reset-knop al een beetje los begon te zitten). De schermafhandeling was erg traag, vooral als het zogenaamde onderwaterscherm aanstond, hetgeen meestal nodig was omdat je anders niet kon zien wat je deed. Als je een 'speciaal' teken in een zin gebruikte, werd onmiddellijk de regelafstand vernaggeld, maar ach… een kniesoor die daarop let. Er waren wel ergere dingen op de wereld, zoals de ligatuur 'ff' die in cursief schrift plotseling rechtop werd gezet…

Iets beters dan WP had ik — voor zover ik wist — nog niet voorhanden gehad. De mogelijkheden van WP leken, vergeleken met die van andere tekstverwerkingspakketten, tamelijk uitgebreid. En, o ja, ik had ook nog wel eens iets gezien van LATEX, tijdens een van mijn eerstejaarscolleges bij mijn informatica-opleiding. Daar had ik me maar niet zo in verdiept, aangezien de LATEX-documenten die ik te zien kreeg er niet echt enthousiasmerend uitzagen. Alle met vrijwel dezelfde standaardopmaak: aanpassen aan eigen wensen was misschien wel mogelijk, maar dan kennelijk toch zeer moeilijk. En over de standaardopmaak had ik zo mijn twijfels. Dictaten met zo veel witruimte om formules, dat een bladzijde uit zo'n dictaat meer leek op een zebrapad dan op een stuk logisch gestructureerde tekst. Een lettertype dat na kopiëren vrijwel onleesbaar werd, en bovendien naar mijn smaak te klein was. Idiote voetnoot-symbolen zoals '†', die mij in eerste instantie vaak deden denken dat een artikel postuum was verschenen. Ik WeePeede dus vrolijk door. Tot die bewuste dag in december 1994…

Drie kwartier — ja u leest het goed: drie kwartier — ben ik bezig geweest met het verrichten van de volgende handelingen:

1. computer resetten
2. DOS laten starten
3. WP opstarten (duurt lekker lang)
4. bestand inlezen (gaat ook niet echt snel)
5. voorzichtig proberen om met de cursor de tweede bladzijde van mijn artikel te bereiken, en…
6. WP vast laten lopen en alle genoemde stappen weer van voren af aan.

Toen ik eindelijk doorhad dat ik eerst de kolommen uit moest zetten, toen de indelingsstijlen, toen de kolommen weer aan moest zetten, en vervolgens weer sommige indelingsstijlen, of iets van die strekking, wist ik eindelijk mijn tekst nog enigszins te redden. Was dat niet gelukt, dan had ik slechts een onbruikbaar WP-bestand gehad, waarvan geen leesbare ASCII-versie bestond. Ik had nog net tijd om bladzijde drie tot en met zeven uit te draaien (bladzijde één en twee van het document, waarop WP steeds vastliep, had ik op dat moment niet eens nodig), naar de schuur te rennen, mijn fiets mee te grissen, te oefenen voor de Tour

de France en op tien seconden nauwkeurig mijn trein naar Almere te halen. In die trein heb ik vervolgens nog minstens een kwartier zitten uitblazen, uitzweten, afkoelen en bedenken wat voor een indruk ik in Almere zou maken met mijn zeven dagen ongeschoren gelaat. Toen ik 's avonds in Arnhem terugkeerde, deelde men mij mee dat de redactie net had besloten om de deadline een weekje te verzetten…

Er zijn grenzen aan wat een mens kan verdragen. WP begaf zich daar nu ver buiten en ik was niet van plan nog langer te volgen. Nu was ik het echt zat. En toen kwam de welhaast perfect getimede brief van Frans Goddijn. Hij nodigde me van harte uit om eens langs te komen om te zien hoe TEX werkt en het eventueel te kopiëren. Ik hapte uiteraard onmiddellijk toe. Alles best, als WP maar zo spoedig mogelijk van mijn vaste schijf kon worden geknikkerd. Ik was bereid enige tijd uit te trekken om me in TEX te verdiepen; per slot van rekening had ik ook jaren de tijd gehad om WP — voor zover mogelijk — meester te worden.

TEX zou zich moeten bewijzen: ik wist dat TEX goed was in de details (afbreken, uitlijning enzovoort) maar ik stel vooral hoge eisen aan de globale vormgeving. Daarin ben ik bovendien eigenwijs: ik wil het zo hebben zoals ìk het wil, en niet zoals Knuth, Lamport, Eijkhout of wie dan ook het heeft bedacht.

Wat ik wilde

Testcase zou uiteraard het soort document worden waarop ik met WP was vastgelopen, en waarvan ik er in het komende half jaar nog een stuk of tien zou moeten produceren. Dat waren uiteraard geen eenvoudige briefjes: die kan (zelfs) WP nog wel aan. Het ging om een voorstel voor nieuwe reglementen van een muziekvereniging, dat per hoofdstuk zou worden gepresenteerd, met daarbij steeds een toelichting en een enquêteformulier. Dit voorstel stelde mij voor de nodige typografische problemen.

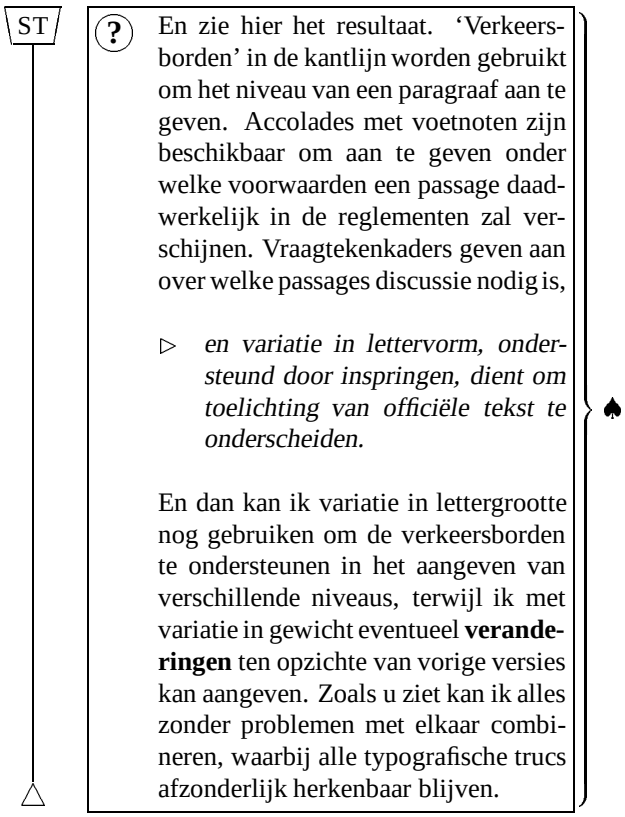
De indeling van het voorstel moest onafhankelijk zijn van die van de brieftekst of het artikel waarin het was verwerkt. Duidelijk herkenbaar moest ik kunnen aangeven welke delen van het voorstel in ieder geval ter discussie stonden, waar de musici moesten kiezen uit verschillende mogelijkheden, wat voorsteltekst was en wat toelichting enzovoort. Sommige delen van het voorstel zouden alleen in de definitieve tekst verschijnen als aan bepaalde voorwaarden werd voldaan; dit moest ik handig aan kunnen geven. Sommige delen waren bedoeld als tekst voor de statuten, anderen als tekst voor het huishoudelijk reglement, en anderen als tekst van een bestuursbesluit. Al deze delen waren geordend op basis van onderwerp met behulp van kopjes en alineanummering, maar er moest dus ook nog iets worden gevonden om het 'niveau' van een stuk tekst duidelijk aan te geven (statutair, reglementair of bestuurlijk).

Al met al had ik minstens vier typografische middelen nodig om een en ander aan te kunnen geven (niveau, al dan niet voorwaardelijk, al dan niet ter discussie, voorstel versus toelichting), waarvan de eerste drie op alle mogelijke manieren moesten zijn te combineren. Bovendien hield ik graag nog wat achter de hand om veranderingen ten opzichte van oudere versies aan te kunnen geven. Aan het variëren van lettertype, -gewicht en -vorm had ik dus niet genoeg, en variatie in lettergrootte is zonder aanvullende aanduidingen voor grotere stukken tekst niet duidelijk. Gezien de grote typografische problemen was het wenselijk dat ik een en ander in LATEX zo kon opzetten, dat ik met de opmaak naar hartelust kon experimenteren zonder steeds wijzigingen in het document zelf aan te moeten brengen. Bij WP ging dat niet zo best: het kon in principe wel een beetje met behulp van 'stijlen', maar bij intensief gebruik daarvan bleek WP het niet meer te zien zitten en verstrikt te raken in eindeloze apathie.

Na met enige moeite 4allTEX te hebben geïnstalleerd, haalde ik mijn oppervlakkige kennis van LATEX weer op. Die had ik destijds opgedaan bij dat eerstejaars-college, maar bleek voor mijn eisen volstrekt onvoldoende. En zo kwam het dat ik in het TEXbook dook alvorens ooit één eigen LATEX-document op eigen kracht te hebben geproduceerd.

Wat ik kreeg

Naarmate ik vorderde, werd mij steeds duidelijker waarom al die LATEX-documenten er hetzelfde uitzien. Gemakkelijk was het niet om je eigen stijl te programmeren. Maar langzamerhand raakte ik steeds meer onder de indruk van de mogelijkheden. Toen ik zover was gevorderd met het uitprogrammeren van mijn opmaak dat ik de eerste ruimschoots acceptabele afdrukken had weten te maken, realiseerde ik me dat ik op een heel andere manier was gaan denken. Toen ik WP gebruikte, ging ik uit van de mij bekende mogelijkheden en gebruikte daarvan wat van pas kwam. Sinds ik TEX gebruik, bedenk ik eerst wat ik wil hebben en vervolgens zorg ik dat ik dat voor elkaar krijg. De mogelijkheden om je eigen creativiteit in je TEX-opmaak te stoppen zijn voor een WP-gebruiker werkelijk ongekend!



In de tekst van het voorstel worden geen typografische middelen aangeduid. Macro's als `\optie`, `\afhankelijk`, `\rartikel`, `\begin{toelichting}` duiden de functie van diverse passages aan. Deze macro's worden allen gedefinieerd in de style file `statuten.sty` die ik bij het document heb geschreven. Wijzigingen in de opmaak worden eenvoudig aangebracht door de style file bij te werken; aanpassingen in de voorsteltekst zijn daartoe zelden nodig.

Oorspronkelijk bevatte `statuten.sty` grote hoeveelheden complexe macro's waarin de opmaak van het voorstel in TEX was uitgeprogrammeerd. Onder invloed van het enthousiasme van Frans Goddijn, die enkele van mijn 'uitvindingen' onmiddellijk in zijn eigen teksten toe wilde passen, besloot ik om zoveel mogelijk uit `statuten.sty` te halen en onder te brengen in afzonderlijke style files. De afzonderlijke style files zouden dan gegeneraliseerde versies van de oorspronkelijke macro's gaan bevatten, zodat ze ook buiten de genoemde reglementen goed bruikbaar zouden zijn.

Sindsdien heb ik weinig tijd meer gehad voor iets anders dan TEX. Frans had bedacht dat ik mijn style files mooi in de MAPS kon presenteren. Tussen het uitwerken van de gegeneraliseerde macro's en het debuggen door moest ik nu dus ook aan documentatie, demonstratie en presentatie werken. Het resultaat is inmiddels te vinden op FGBBS en in twee artikelen in deze MAPS: *The Scenario — in Three Versions* en *HH Gets Carried Away*. Het eerste beschrijft onder meer de macro's om verkeersborden en accolades naast de tekst te zetten. Het tweede presenteert onder meer de style files die ik gebruik voor mijn voetnoten en de omcirkelde vraagtekens. De macro die ik gebruik om vraagtekenkaders te zetten gaat nog niet zo handig om met witruimte, maar is desondanks al beschikbaar in de file `hhparmrk.sty` die in *The Scenario — in Three Versions* wordt gepresenteerd.

Hoe het verder gaat

WP wordt op mijn systeem eigenlijk alleen nog maar tot leven gewekt wanneer nog niet van WP bevrijde slachtoffers (meest familieleden) bij mij aankloppen voor mentale en praktische bijstand.

Mijn studie bezwijkt haast onder mijn TEXitis. Ik ben veel meer tijd kwijt aan TEX dan ik ooit in WP heb gestopt, maar dan hebben we het hier wel over een hack-verslaving, iets dat mij geregeld overvalt en deze keer kennelijk op TEX is gericht.

De teksten met het voorstel voor de reglementen van de muziekvereniging zijn tegenwoordig steeds in een mum van tijd gereed. De meeste tijd gaat nu zitten in de inhoud en in de tweede plaats in de afwerking: de af en toe noodzakelijke kunstgrepen om iets nog net op één bladzijde te proppen (teneinde de kopieerkosten te beperken). Met de globale vormgeving hoef ik me nauwelijks nog bezig te houden: die heb ik immers al vastgelegd in `statuten.sty` en wordt met een simpel `\usepackage{statuten}` in elk document tot leven gewekt. Met WP was ik aan inhoud en afwerking net zo veel tijd kwijt, maar desondanks ging de meeste tijd zitten in het tussenliggende traject: de complexe globale vormgeving, waar het gulzige WYSIWYG-WP zich vaak in verslikte in zijn onstuimige haast om resultaat te tonen. Ongelukken die zich in mijn WP-tijd bij elk hoofdstuk van mijn voorstel voor nieuwe reglementen meerdere malen herhaalden.

Voor incidentele teksten heb ik inmiddels voldoende TEX-ervaring om een WP-achtige manier van werken te kunnen simuleren, door het veelvuldige gebruik van opmaakmacro's in plaats van macro's die de logische structuur van het document aanduiden. Desondanks kost het typen van een kort briefje in TEX meer tijd dan in WP: vaak is een tweede compilatie-ronde nodig om wat kleine foutjes te verhelpen die na de eerste ronde aan het licht kwamen. Ik ben echter zo gehecht geraakt aan de meer verfijnde vormgeving door TEX, dat ik die geringe extra inspanning er wel voor over heb.

♠ Dit soort voetnoten gebruik ik om aan te geven hoe het opnemen van een passage in de reglementen afhankelijk is van het opnemen van andere passages.

HH Gets Carried Away

hhmuf, hhflxbox and hhcount

Herman Haverkort

herman@fgbbs.iaf.nl

March 1995

Abstract

This article presents hhmuf's multinotes: special cheery footnotes to be used in special situations, including so-called 'forbidden environments'. Then it presents hhflxbox's self-scaling frames: encircling macros are provided but you can define enwhatevering macros yourself by means of the macros provided by hhflxbox. Finally hhcount is presented: macros to handle simple and composite counters in a fancy way.

Keywords: circling, counter, dice, footnote, forbidden environment, frame, index

In this article I present some of the features of a few packages I wrote recently. However this presentation is far from complete. A more detailed manual and the packages itself can be obtained from FGBBS.[1] To use the packages described here additional packages are needed. You will get everything you need if you request the file `hh.arj` from FGBBS. I will try to submit the hh packages to CTAN as well. Note that LaTeX2.09 versions are not available.

1 Some Features of the hhmuf Package

1.1 Reusable Footnotes and Recycled Markers

Suppose you have to typeset some tables containing many entries which are amplified by footnotes outside the table. Several entries, possibly but not necessarily in the same table, refer to the same footnote. If references to the same footnote appear in different tables on different pages, the footnote text should be set on all pages involved, while the footnote marker should be the same each time the footnote is set. The first to avoid unnecessary turning over, the second to avoid confusion.

Typesetting such tables is not very easy in basic LaTeX. A relatively easy way to do the job is probably as follows:

1. first get markers for all the footnotes and define macros to set the footnote markers;
2. then define a macro `\tablenotes` which typesets the footnotes;
3. then typeset the tables, using the macros defined in the steps mentioned above;

Here is some example input:

```
% step 1:
\newcommand\getfootnotemarker[1]{%
  \stepcounter{footnote}%
  \newcounter{#1}%
  \setcounter{#1}{\value{footnote}}%
  \expandafter\newcommand\csname #1\endcsname
    {\footnotemark[\value{#1}]}}
\getfootnotemarker{notea}%
\getfootnotemarker{noteb}%
\getfootnotemarker{notec}%

% step 2:
\newcommand\tablenotes{%
  \footnotetext[\value{notea}]{%
                    First example footnote}%
  \footnotetext[\value{noteb}]{%
                    Second example footnote}%
  \footnotetext[\value{notec}]{%
                    Third example footnote}}

% step 3:
\tablenotes
\begin{center}
\begin{tabular}{|l|l|}%
\hline
  name       & amount in \$ \\
\hline
  Achterberg & 100          \\
  Bosman     & 150\notea    \\
  Evers      & 125\noteb    \\
  Gerritsen  & 145          \\
  Hooier     & 170\notec    \\
  Jansen     & 165\notea    \\
\hline
\end{tabular}
\end{center}
```

[1] FGBBS — tel. (085) 21 70 41

And the resulting output:

name	amount in $
Achterberg	100
Bosman	150[2]
Evers	125[3]
Gerritsen	145
Hooier	170[4]
Jansen	165[2]

This article is too short to demonstrate it, but when typesetting multiple tables this way problems are likely to arise. If you typeset another table on the same page, calling `\tablenotes` again will cause the footnote texts to be typeset twice on the same page. If you typeset another table on another page, *all* table footnote texts will be typeset on that page, even if the table which is on that page does not refer to all of them.

The hhmuf package solves these problems, although the solution of the twice-on-the-same-page problem may be buggy in rare contexts. If you use the hhmuf package you can replace the previous listing by the following:

```
\mufhire note1:{First example footnote}%
\mufhire note2:{Second example footnote}%
\mufhire note3:{Third example footnote}%
%
\begin{center}
\begin{tabular}{|l|l|}%
\hline
  name       & amount in \$      \\
\hline
  Achterberg & 100               \\
  Bosman     & 150\muf note1:{}  \\
  Evers      & 125\muf note2:{}  \\
  Gerritsen  & 145\muf:{Example
          of an incidental note} \\
  Hooier     & 170\muf note3:{}  \\
  Jansen     & 165\muf note1:{}  \\
\hline
\end{tabular}
\end{center}
```

And get this result:

name	amount in $
Achterberg	100
Bosman	150△
Evers	125⊖
Gerritsen	145×
Hooier	170♣
Jansen	165△

[2] First example footnote
[3] Second example footnote
[4] Third example footnote
△ First example footnote
⊖ Second example footnote
× Example of an incidental note
♣ Third example footnote
[5] which occurs frequently in my ever changing text

You will note that `\mufhire` *label*`:{`*footnote text*`}` is used to define footnotes. Its opponent is, of course, `\muffire` *label*`:`, which undefines footnotes, while `\muf` *label*`:{}` is used to set previously defined footnotes. As shown in the example, `\muf:{`*footnote text*`}` can be used to set incidental footnotes. `\muf:{`*footnote text*`}` actually acts as an abbreviation for hiring, typesetting and firing an incidental note. Thus it is well-suited for setting normal in-text footnotes.

hhmuf does not use common footnote markers. Most common sets of symbols have a well-defined order which makes them ill-suited for hhmuf since the hhmuf macros do not respect that order. While defining footnotes, markers are assigned in turn. Thus there is no need to restart footnote numbering every chapter or every page, because you never run out of markers, unless you hire a lot of them without ever firing any. Restarting footnote numbering does not make sense anyway because there is no such thing as a *first* marker.

The set of markers used by hhmuf can be fully specified by the user, either by selecting one of the predefined sets or by compiling a new one. The predefined sets are the following:

set name	markers included
black	• ♦ ▼ ♣ ■ ▲ ◀ ♠
circlox	⊘ ⊙ ⊠ ⊕ □ ⊛ ⊟ ⊗ ⊚ ⊡ ⊖ ⊞
fuss	∗ ♢ ⊛ ⋇ ♯ ⋆ ♣ ℵ ∞ ♁
geometry	♦ □ ▼ △ ■ ◁ ▲ ♢ ◀ ▽
misc	♠ △ ⊖ ♣ × ♢ ⊗ ≀ ⊙ ∨ ⊕ ∞ ⋆ ⊘ + ◁ ⊤ • ▽
music	♯ ♭ ♮
strokes	⊤ × ⅄ + ⊻ ⊼ ⊼ ≀

For details see the documentation in the package file.

1.2 The Forbidden Environment Problem

Suppose I typed several pieces of text that have to be typeset all in the same special way. For that purpose I (re)defined an environment `specialtext`:

```
\renewenvironment{specialtext}%
    {\par$\bigtriangledown$\par}%
    {\par$\bigtriangleup$\par}
```

▽

If I have a piece of text like this paragraph, which refers to a footnote,[5] this should cause me no problems.

△

Then I defined a package option in the package I used to typeset my document. If I specified that option when loading the package, then `specialtext` would be defined as follows:

```
\renewenvironment{specialtext}%
    {\par\setbox0\vbox\bgroup\hsize5cm\relax}%
    {\egroup
     \begin{center}\fbox{\box0}\end{center}}
```

Now problems arose. I typeset the same text again:

> If I have a piece of text like this paragraph, which refers to a footnote,[6] this should cause me no problems.

The paragraph is shown framed all right, but something went wrong with the footnote. With the new definition of `specialtext`, the footnote suddenly appears in a 'forbidden' environment and therefore it actually disappears. Although the in-text marker is typeset, there is no note at the foot of the page.

While writing this article I discovered Kresten Krab Thorup's style file `ftn.sty`,[7] which attempts to solve this problem but does so quite buggily. Footnotes disappear when forbidden environments are nested. When multiple footnotes are type-set in forbidden environments, footnotes are repeated and their numbering is wrong. A modified version of `ftn.sty` exists (by Zdenek Wagner), which solves the repetition problem correctly, and suppresses incorrect numbers by omitting them: at least that is what I got. I tried to contact Krab Thorup about making `ftn.sty` more robust, but did not succeed so far. Nevertheless Krab Thorup's style file contained some very useful ideas, which I combined with my own ideas to construct a set of macros which seem to be quite robust. I will now show you the result: how easy it is to use hhmuf's footnotes in forbidden environments.

The kind of unpleasant surprises presented above is easy to prevent when typesetting footnotes with `\muf` instead of `\footnote`, like in:

```
\begin{specialtext}
If I have a piece of text like this paragraph,
which refers to a footnote\muf:{which occurs
frequently in my ever changing text}, this
should cause me no problems.
\end{specialtext}
```

When using the first definition of `specialtext`, this yields:

▽

If I have a piece of text like this paragraph, which refers to a footnote,[◇] this should cause me no problems.

△

In the following example the second definition of `specialtext` is used, but I added one line of code to 'protect' the environment:

```
\renewenvironment{specialtext}%
    {\par\setbox0\vbox\bgroup\hsize5cm\relax}%
    {\egroup
     \begin{center}\fbox{\box0}\end{center}}
\mufoff[specialtext]
```

And here is the result:

> If I have a piece of text like this paragraph, which refers to a footnote,[⊗] this should cause me no problems.

Without changing the text in my document, I could redefine `specialtext` to use forbidden environments in such a way that my footnotes did not disappear. `\mufoff` did the job.

1.3 Shortcomings: `minipage` Fans Beware!

hhmuf does not support `minipage`s yet. If `\muf` is used in a `minipage` environment, the footnote will be placed at the foot of the 'master page' instead of under the `minipage`!

2 The hhflxbox Package

hhflxbox contains a number of boxing macros. The kernel consists of `\iframe`, which boxes things and sets self-scaling frames around, and `\sframe`, which sets more complex self-scaling and -stretching frames. Besides hhflxbox provides the encircling macros `\ringbox`, `\bellybox` and `\outringbox` (which use `\iframe`), the macros `\sepbox` and `\separbox`, which set empty space around boxes, and `\broadbox`, which boxes its argument in a `\vbox` of which the width is the line width minus some specified value.

2.1 `\sepbox` and `\separbox`

For introduction of `\sepbox` and `\separbox` it is convenient to introduce `\bellybox` first. `\bellybox` is one of the hhflxbox macros which can be used to encircle things, for example ③, which is set with: `\bellybox :{3}`.

You probably notice that the circle around the digit is somewhat tight. This problem can be solved by putting a `\separbox` around the digit, like in `\bellybox`

[7] Available at CTAN as `macros/latex209/contrib/misc/ftn.sty`

⊗ which occurs frequently in my ever changing text

`:{\separbox{1pt}{3}}`, which yields: ③. Actually `\separbox{`*dimension*`}{`*stuff*`}` puts *dimension* wide empty space around *stuff* on all sides.

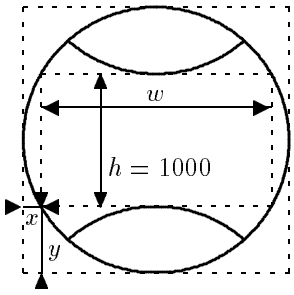
A more general form is:
`\sepbox(`*leftspace*`,`*topspace*`,`*rightspace*`,`*bottomspace*`){`*stuff*`}`
which adds empty spaces of the specified widths to the sides of the box containing *stuff*.

2.2 `\iframe`: Isomorphous Frames

`\iframe` is only a frame drawing *tool*: it does not draw frames itself but it can take care of the proper positioning and scaling of frames drawn by other macros. To explain the functioning of `\iframe` it is probably best to give an example of the development of a framing macro using `\iframe`.

Suppose we want to set self-scaling frames which have the following shape:

then we could imagine a box-shaped area in the frame which will contain the frame's contents (the inner dashed box in the figure below). Also we could imagine a box surrounding the frame (the outer dashed box in the figure).



Since `\iframe` expects the inner box height to be 1000 times the `\unitlength`, all dimensions have to be chosen so that the inner box height equals 1000 indeed. Then `\iframe` can scale the frame by setting the `\unitlength`. Furthermore `\iframe` expects the lower left corner of the outer box to have coordinates $(0,0)$. Taking these expectations in account we can design a macro which draws the frame:

```
\newcommand\sillyshape{%
  \begin{picture}(2000,2000)
    \thicklines
    \put(1000,1000){\arc(0,1000){360}}
    \put(1000,-498){\arc(662,749){83}}
    \put(1000,2498){\arc(-662,-749){83}}
  \end{picture}}
```

(`\arc` is defined in the curves package by I.L. Maclaine-cross.) Now we can define a silly shape framing macro by defining `\sillyframe` as: `\iframe \sillyshape(`x`,`y`){`w`}{0pt}\ifrch\ifrcv :{#1}`. Actually we will set `#1` in a `\sepbox (0pt,2pt,0pt,2pt){#1}` to prevent the frame from touching its contents. In the above example we have $x = 134$, $y = 500$ and $w = 1732$, so we write:

```
\newcommand\sillyframe[1]{%
 \iframe\sillyshape(134,500){1732}{0pt}%
  \ifrch\ifrcv:{\sepbox(0pt,2pt,0pt,2pt){#1}}}

Now we can put \sillyframe{all}
\sillyframe{sorts} \sillyframe{of}
\sillyframe{things} in silly frames.
```

Now we can put (all) (sorts) (of) (things) in silly frames.

Note that I do not claim this kind of silly frame to be good-looking: it is just an example.

The dimension `0pt` in the example above determines the minimal height of the silly frame's inner box. Sometimes it is necessary to define it because LaTeX's picture environment suppresses small line segments.

The macro `\ifrch` determines what should be done if the frame's contents width/height ratio is too small. By specifying `\ifrch` we instruct `\iframe` to center the contents. Instead of `\ifrch` we could have specified `\ifrl` or `\ifrr` to have the contents flush left or right.

The macro `\ifrcv` determines what should be done if the frame's contents height/width ratio is too small. `\ifrcv` yields vertical centering, while `\ifrt` and `\ifrb` yield top and bottom flushing.

If we put frames around for example page numbers, then the self-scaling properties of isomorphous frames may have an unpleasant result: numbers of the same type, like page number (21) and page number (25), might get differently sized frames because of their different natural sizes. This can be solved by redefining `\sillyframe` to specify a *unit name*, since all things typeset with the same unit name get equally sized frames. The unit name, for example `pagenr`, should be placed between the vertical alignment specification and the colon, like in:

```
\newcommand\pagenrframe[1]{%
  \iframe\sillyshape(134,500){1732}%
    {0pt}\ifrch\ifrcv pagenr:{%
      \sepbox(0pt,2pt,0pt,2pt){#1}}}

framed numbers like \pagenrframe{\oldstylenums
{21}} and \pagenrframe{\oldstylenums{25}}.
```

which yields (after compiling our document twice):

framed numbers like (21) and (25).

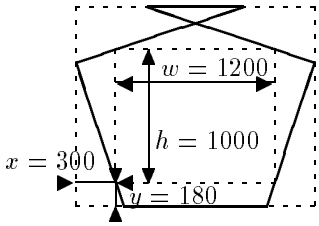
However, this is not fully satisfactory yet: now the frames are equally sized but the first frame is positioned higher than the second. This is no bug, it is a feature. No really, it is! It is, however, a sometimes unwanted feature. The solution is using `\lcenter` to center the frames on their line, like in:

```
pages \lcenter{\pagenrframe{\oldstylenums{21}}}
and \lcenter{\pagenrframe{\oldstylenums{25}}}
```

resulting in:

pages 21 and 25

As a final example of isomorphous frames, consider the following framing macro. Note that the inner box height is 1000 again, as expected by `\iframe`.




```
\newcommand\jarshape{%
  \begin{picture}(1800,1500)
    \thicklines
    \put(360,0){\line(-1,3){360}}
    \put(0,1080){\line(3,1){1260}}
    \put(540,1500){\line(1,0){720}}
    \put(1440,0){\line(1,3){360}}
    \put(1800,1080){\line(-3,1){1260}}
    \put(360,0){\line(1,0){1080}}
  \end{picture}}

\newcommand\jarframe[1]{%
 \iframe\jarshape(300,180){1200}{10pt}%
  \ifrch\ifrb:{\separbox{1pt}{#1}}}
```

2.3 `\ringbox`, `\bellybox` and `\outringbox`: Encircling

`\ringbox{`*optional unit name*`}:{`*stuff*`}` sets a circle around *stuff*. The specification of a unit name is optional; its use is explained above.

`\outringbox` is very much like `\ringbox`, but the following example demonstrates their difference:

```
1\ringbox:{2}3 and 1\outringbox:{2}3
```

yields:

1②3 and 1②3

If `\ringbox` is used, the circle contributes to the width, height and depth of the result. If `\outringbox` is used, the circle does not contribute any width, height or depth, so that the text is typeset as if the circle were not present and the circle were added after typesetting the text.

The result of `\bellybox` is a circle which contributes a bit to the dimensions of the encircled result but also sticks out a bit (by 10 percent of its radius to be sort of exact). So `\bellybox` is an intermediate form of `\ringbox` and `\outringbox`.

2.4 `\sframe`: Stretchable Frames

Putting `\sframe` to good use is a rather complex task. `\sframe` assembles user-defined frame components which actually are macros which set the values of a box and several dimension registers. Therefore I decided to give only an example of what can be achieved in this article; for explanation see the manual and demo files available at FGBBS.

If one has defined suitable macros `\fancycolumn` and `\fancytympan`, one can define:

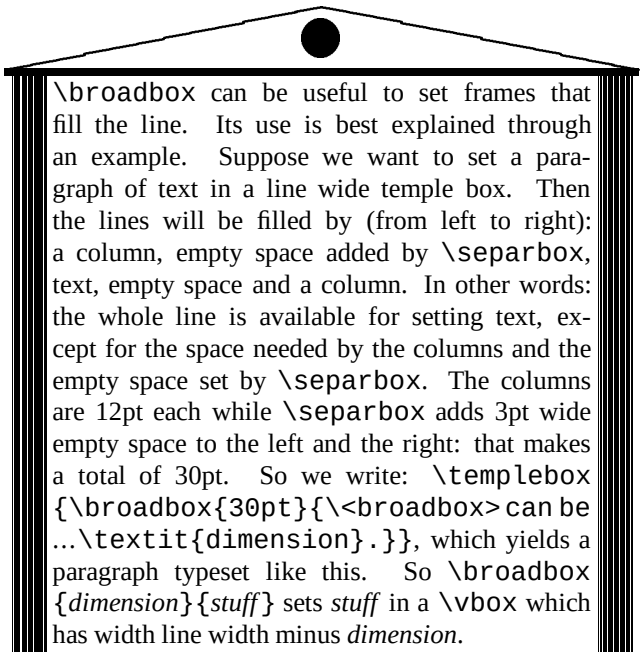
```
\newcommand\templebox[1]{\sframe
  [1]\fancycolumn [2]\fancytympan
  [1]\fancycolumn [-]\-%
  {\separbox{3pt}{#1}}}
```

which should be read as: after 3pt wide empty space is set around `#1`, first add columns to the left and the right, second put a tympan on top of the result, and never put anything at the foot. Then typing this:

```
\templebox{hello there!} and \templebox{%
  \vbox{\hbox{b}\hbox{y}\hbox{e}}}
```

will yield:

2.5 `\broadbox` for Setting Line Wide Frames



`\broadbox` can be useful to set frames that fill the line. Its use is best explained through an example. Suppose we want to set a paragraph of text in a line wide temple box. Then the lines will be filled by (from left to right): a column, empty space added by `\separbox`, text, empty space and a column. In other words: the whole line is available for setting text, except for the space needed by the columns and the empty space set by `\separbox`. The columns are 12pt each while `\separbox` adds 3pt wide empty space to the left and the right: that makes a total of 30pt. So we write: `\templebox {\broadbox{30pt}{\<broadbox> can be ...\textit{dimension}.}}`, which yields a paragraph typeset like this. So `\broadbox {`*dimension*`}{`*stuff*`}` sets *stuff* in a `\vbox` which has width line width minus *dimension*.

2.6 Environment Versions

Some of the macros defined in hhflxbox are available as LaTeX environments. For example: instead of `\broadbox{30pt}{`*text to be boxed*`}` one could also use `\begin{broadboxed}{30pt}` *text to be boxed* `\end{broadboxed}`. Similarly one could use the environments `sepboxed`, `separboxed` and `sframed` instead of the macros `\sepbox`, `\separbox` and `\sframe`.

Actually I have to confess something: I lied to you about the typesetting of the section about `\broadbox`. I did it with:

```
\newenvironment{templeboxed}{%
  \begin{sframed}%
    [1]\fancycolumn [2]\fancytympan
    [1]\fancycolumn [-]\-
    \begin{separboxed}{3pt}
      \begin{broadboxed}{30pt}
}{%
      \end{broadboxed}%
    \end{separboxed}%
  \end{sframed}%
}

\begin{templeboxed}%
  \<broadbox> can be useful to set frames that
       :        :        :        :        :
  width line width minus \textit{dimension}.
\end{templeboxed}
```

I hope you will forgive me my cheating. I mean… without using the environments typesetting verbatim stuff is so troublesome…

3 Some Features of the hhcount Package

3.1 Simple Number Formatting

Let us start by summarizing the simple number formatting macros which are provided by hhcount:

example input	corresp. output	other example output
`\fctabdigit{2}`	2	29
`\fcolddigit{2}`	2	29
`\fcloweralpha{2}`	b	cc
`\fcbigalpha{2}`	B	CC
`\fcsmallalpha{2}`	B	CC
`\fclowerroman{2}`	ii	xxix
`\fcbigroman{2}`	II	XXIX
`\fcsmallroman{2}`	II	XXIX
`\fcbigromanlined{2}`	II	XXIX
`\fcsmallromanlined{2}`	II	XXIX
`\fcbigdice{2}`	⚁	⚀⚀⚀⚀⚁
`\fcsmalldice{2}`	⚁	⚀⚀⚀⚀⚁
`\fcbigscore{2}`	𝍡	𝍠𝍠𝍠𝍠𝍠𝍡
`\fcsmallscore{2}`	𝍡	𝍠𝍠𝍠𝍠𝍠𝍡
`\fcfnsymbol{2}`	†	

The next step in complexity are number formatting macros which give context-dependent output. This is implemented by using the *context switches* `\if@fcoldstyle` and `\if@fcsmall`, which are set by context switching macros like `\fcinheading` and `\fcintext`. We say that a context switching macro is active if it was the last one to affect the context switches.

example input	output when `\fcinheading` is active	output when `\fcintext` is active
`\fcdigit{14}`	14	14
`\fcalpha{14}`	N	N
`\fcroman{14}`	XIV	XIV
`\fcromanlined{14}`	XIV	XIV
`\fcdice{14}`	⚁⚁⚀	⚁⚁⚀
`\fcscore{14}`	𝍠𝍠𝍡	𝍠𝍠𝍡

By default `\fcinheading` is active; `\fcintext` is active when using `\ref` or `\pageref` (those two macros are redefined by hhcount).

3.2 How to Define Composite Counters

I will now try to give an impression of the way in which composite counters can be defined using hhcount. However, this is *not* a manual. After reading the following paragraphs you may be able to hack a composite counter together yourself, by imitating what is done below and experimenting with some small modifications of your own. If you want to be taught how to use hhcount efficiently and effectively, then you should read the manual.

Suppose we want to set up a three-level section numbering system for some sub-document in another document, for example the rules of a club embedded in some booklet about that club. The section numbers should be composed from the values of three (sub-)counters: `ruleschapter`, `rulessection` and `rulesparagraph`. Chapter numbers should be represented by capital alphabetic characters; elementary section and paragraph numbers by arabic digits. What should be done?

First we select a *series identifier* for our composite counter. Series identifiers are natural numbers which are assigned to composite counters. Each composite counter should be assigned a unique identifier. Because identifiers `1` to `8` and `12` are reserved for common purposes we select `9` for our rules section numbers.

Then we define a macro which expands to the series identifier:

```
\def\rulesseries{9}
```

Next we specify how the three sub-counters are to be combined:

```
\combinecounters\rulesseries{%
  \\{ruleschapter}%
  \\{rulessection}%
  \\{rulesparagraph}}
```

And finally we define how the counter is to be formatted:

```
\setcounterformat\rulesseries{#1-#2-#3}{%
  \fcorfinally
  % capitals for chapter numbers:
  \fcformat{#1}{\fcalpha}%
  % digits for section numbers:
  \fcformat{#2}{\fcdigit}%
```

```
% a period to separate section and paragraph
% numbers:
\fcformat{#3}[.]%
% digits for paragraph numbers:
             {\fcdigit}%
\fcordespair}
```

If you want to understand the definition above, read the manual.

3.3 The Result

Now the composite counter can be accessed by the macros `\theruleschapter`, `\therulessection` and `\therulesparagraph`, which give results like: 'A', 'A2' and 'A2.3'. The macros `\stepcounter{ruleschapter}`, `\stepcounter{rulessection}` and `\stepcounter{rulesparagraph}` can be used to step the counter.

When `\fcinheading` is active, rules paragraph numbers will be set like 'A2.3', but when `\fcintext` is active, the same number will be set like 'A2.3'.

More complex distinctions in representation of counters are possible. hhcount provides a set of macros which can be used in the last argument of `\setcounterformat`. Those macros enable definition of counters which are set like 'A2.3' in headings and like 'section A2, par. 3' in text etc. For details see the manual.

3.4 hhcount and makeindex

Composite section numbers like 'A2.3' cannot be handled by the `makeindex` program. Besides `makeindex` has problems with sorting alphabetic numbers since it cannot determine whether or not it are roman numbers. hhcount provides a way to get around these problems.

All composite numbers defined by hhcount constructs are internally represented by a sequence of natural numbers, separated by hyphens and embedded in a macro call. A typical example is `\fancycounter 9-1-2-3-!`. The first number represents the series identifier (9 in the example), while the following numbers represent the values of the relevant sub-counters.

hhcount provides macros `\initfancycounters`, `\indextolabels` and `\indextopages`. The first redefines the section and page numbering systems to use hhcount's composite counters. `\indextolabels` sort of redefines `\index` to use the redefined section numbers and strip the `\fancycounter` and the `-!` off the composite counter representation. `\indextopages` does the similar thing for page numbers. In both cases the result is a sequence of natural numbers, separated by hyphens, which can be handled perfectly well by `makeindex`. By embodying the appropriate definitions in your index style (`.ist`) file `makeindex` will undo the stripping after sorting the page or section numbers, so that your index entries will still be typeset as defined by means of hhcount macros. Thus section numbers like 'A2.3' can be used for references in the index. Inserting equation, table and figure numbers in the index is just as easy. It is even possible to have different kind of composite numbers in the same index, for example page as well as section numbers, because the series identifiers are not stripped off so that it remains possible to determine the proper series and formatting of each composite number. For details see the manual.

3.5 Bugs and Deficiencies

Class files tend to make the TEX compiler show on your terminal which chapter of your book or report is being processed. Error messages often contain the page number. When using hhcount there is a chance that the chapter and page numbers shown on your terminal look weird: you will be shown the internal representation of your counter (`\fancycounter 9-1-0-0-!` for example). This is caused by an incorrect timing of macro expansion: in this case `\fancycounter` is expanded too late, that is: not at all.

Late expansion with hhcount is typically a problem with error and other messages: I would be highly surprised if someone discovers something like `\fancycounter 9-1-0-0-!` outside verbatim environments in a typeset document. However, when compiling your document you might run into early expansion, which causes severe errors. With the latest version of hhcount this problem does not seem to emerge in 'usual' contexts; however I am not sure.

Front matter, appendix and back matter peculiarities (with respect to page and section numbering) are not automatically supported by hhcount. Class files are too different in that respect. If hhcount is to be used to handle the section and page numbering in documents containing front matter and appendices, it would probably be best to incorporate hhcount in the class file, instead of loading it as an additional package.

3.6 Gamesters Page Numbers

The following redefines the `page` counter so that page numbers will be set as dice (I designed this for a gamesters society):

```
\def\fcpageseries{12}
\combinecounters\fcpageseries{\\{page}}
\setcounterformat\fcpageseries{#1}{%
  \fcorfinally
  \fcformat{#1}\fcdice
  \fcordespair}
```

I could not help to give you this as an final example.

The Scenario — in Three Versions

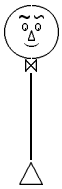
hhparmrk does it

Herman Haverkort & Frans Goddijn

herman@fgbbs.iaf.nl & goddijn@fgbbs.iaf.nl

April 1995

Abstract



During work towards a flexible document as a continuous report on a wide variety of contacts for the Meridian Arts Ensemble in New York, Frans Goddijn felt the need to tag and mark certain paragraphs for specific groups of readers. Herman Haverkort wrote a package for LaTeX2ε, hhparmrk, which facilitates this by offering the possibility to set various signs next to paragraphs. This article presents hhparmrk, gives examples of its use and a short manual. For the hackers among us some of the TeXnical tricks involved behind the scenes are glanced at. ⊙

Keywords: paragraph, mark, distribution, select

During the process of organizing concerts for a delightful brass quintet from New York called the Meridian Arts Ensemble I noticed that there is at least *one* aspect about playing all over the world which causes anxiety. Namely, the fact that in many places, many people (are supposed to) look after your interests and it's very hard to keep track of who is doing what.

I used my knowledge of LaTeX to generate reports on all my Meridian activities. Instead of building up a heap of separate emails, notes, letters and memos I wrote or compiled them all as `\sections` and `\subsections` into the same master document, which I entitled THE SCENARIO. With *CorelDraw!* I designed a title page, LaTeX took care of the table of contents and an elaborate index, and gradually I was generating a tight mass of information which I could print out at will and send to the Meridians.

This LaTeX product impressed them, as they'd never before seen such a 'roadmap' of all that was done or not yet done in their interest. This helped them in making the decision to leave a professional booking agent and let me coordinate all further activities in this part of the world! THE SCENARIO quickly grew in size to over fifty pages filled with valuable information about how to initiate contacts with promoters and presenters in concert halls and other venues. The advantage of using LaTeX over some other typesetting or word processing tools is that it's fast and simple to copy plain ASCII emails into it. For instance, the tuba player of the group uses the internet to send me updates for the `tabular` listing their concert dates. The same goes for newspaper reviews and incoming faxes, which I run through a scanner with OCR to transfer it into electronic text. To brighten up the text page, I sometimes create a graphic scan of a hand written quote of musical score or a concert program.

Over the weeks, the book even got its own 'gossip' section, and soon it turned into a kind of family scrapbook, besides building up the more formal data.

By this time, the new print looked so handsome that it was a shame not to use it for a wider range of readers. Other groups could benefit from the many addresses listed in THE SCENARIO, and some fans had heard of its existence and were keen to get a copy of the book with all its *inside information*.

A stylized portrait of Frank Zappa. The Meridian Arts Ensemble is especially renowned for their interpretations of Zappa's compositions, arranged by Jon Nelson.

How to go about this? I would now need three versions of the document. One full version with all info for me and the

⊙ Just a demonstration of hhparmrk's way of bracing and footnoting a paragraph.

Meridians, one slimmed down version where any explicitly confidential text would be left out and one minimal version especially geared towards concert hall programmers with only the basic material (introduction, biographical info, reviews and the like).

Luckily, I had just introduced a young man to LaTeX. I think that every TeX user has made attempts, with more or less success, to convince others of the beauty and pleasures of TeX and I am no exception to this rule. Most of the time, people have no clue what I'm talking about. Sometimes, one buys the 4allTeX CD-ROM and installs it but rarely an avid new user is born. This Herman Haverkort was different: my letter telling him about LaTeX happened to reach him on a friday when he was most bitterly sick of his 'WordPerfect' software and he immediately took action. He got the software from me, installed it, read the TeXbook from screen ignoring all *dangerous bend*-signs and the next week he devoured the printed book itself.

Herman recognized my problem and as he had a similar project at hand, he created a new style file for us, called hhparmrk.

Now I could brand some paragraphs for *exclusive* readership, others for a *circle* of interested readers and the rest was for *wide* distribution. In the 'Wide' version, paragraphs of the other two categories should vanish automatically (actually there is a fourth version, 'Concise', which is a 'Circle' version limited to the very basic information).

First, I took macros for the *disappearing acts* from the comment package but a little later I figured out an easier way, by using a macro with a parameter and never using that parameter! Here is the disappearing act:

```
\newcommand{\LeaveOut}[1]{}
```

`\LeaveOut{Sh*! Wish I hadn't said that}` enables me to put in some lengthy paragraphs that I don't (yet) intend to really use in print, but want to have there in the source file as my own private comments, or I can have TeX ignore portions of text I might want to use later.

In the 'Circle' version, only the *exlusive* texts must disappear and some more or less confidential paragraphs must be marked accordingly. In the 'Exclusive' version, everything is visible but the reader must be able to see what parts will be occluded for others.

Now look at the following new commands. At first they are without any use, later on they get their tasks assigned. Look at them, bland and expressionless like babies, with only their names to distinguish them from other creations...

```
\newcommand{\ForWhom}{}
\newcommand{\Circle}[1]{}
\newcommand{\Exclusive}[1]{}
\newcommand{\EndConcise}{}
```

This is what they get to do in life:
`\ForWhom` will remember for whom the current version is made.
`\Circle` will be a macro with one argument at a time, namely a paragraph that must be left out in the 'Wide' version and marked as 'Circle' in the other versions.
`\Exclusive` will also be a macro with one argument at a time, this time a paragraph that must be left out in the 'Wide' *and* in the 'Circle' versions and marked as 'Exclusive' in the 'Exclusive' version.
`\EndConcise` will normally mean nothing, but the command is placed at a point in the text where it must end with a new page and an index if I want to create a 'Concise' version.

Then the definitions of the different standard versions:

```
\newcommand{\ForExclusive}{
   \renewcommand{\ForWhom}{'Exclusive'}
   \renewcommand{\Exclusive}[1]{%
                     \MarkThisExclusive{##1}}
   \renewcommand{\Circle}[1]{%
                     \MarkThisCircle{##1}}
                          }
```

What happened above is that `ForWhom` will now remember it's for 'Exclusive' use, and both 'Exclusive' and 'Circle' paragraphs are classified as such in the margin.

```
\newcommand{\ForCircle}{
   \renewcommand{\ForWhom}{'Circle'}
   \renewcommand{\Exclusive}[1]{(\ldots)}
   \renewcommand{\Circle}[1]{%
                     \MarkThisCircle{##1}}
                          }
```

What happened above is that `ForWhom` will now remember it's for 'Circle' use. 'Exclusive' paragraphs are ignored and '(...)' is printed in their place, while 'Circle' paragraphs are classified as such in the margin.

```
\newcommand{\Concise}{
   \renewcommand{\EndConcise}{%
       \newpage \printindex \end{document}}
   \renewcommand{\ForWhom}{concise 'Circle'}
   \renewcommand{\Exclusive}[1]{(\ldots)}
   \renewcommand{\Circle}[1]{%
                     \MarkThisCircle{##1}}
                          }
```

What happened above is that `ForWhom` will now remember it's for 'concise Circle' use. 'Exclusive' paragraphs are ignored and '(...)' is printed in their place, while 'Circle' paragraphs are classified as such in the margin. Furthermore, at the place where the 'dummy' macro `\EndConcise` was loitering, it is now told to end the document neatly with a new page and an index.

```
\newcommand{\ForWide}{
   \renewcommand{\ForWhom}{'Wide'}
   \renewcommand{\Exclusive}[1]{(\ldots)}
   \renewcommand{\Circle}[1]{(\ldots)}
                          }
```

The long dull version. Only harmless material is printed, and lots of it, even beyond the `\EndConcise` macro.

```
\newcommand{\MarkThisExclusive}[1]{%
  \begin{trafficsigned*}{\trapbox:}{\small Ex}
    #1\end{trafficsigned*}}
\newcommand{\MarkThisCircle}[1]{%
  \begin{trafficsigned*}{\ringbox:}{\small Ci}
    #1\end{trafficsigned*}}
```

The above two lines control all markings in the text, using new macros and environments which are defined in hhparmrk.

```
This text may be seen by everybody.

\Circle{This text may be seen, marked with a
        circled ‘Ci’, by a certain circle.}

\Exclusive{This text may only be seen, marked
        with a ‘Ex’ in a trapezoid box, by a
        small group of readers exclusively.}
```

I am now able to change the look and size of the document by activating one of the following commands (while commenting out the others):

```
%\ForExclusive
%\ForCircle
%\ForWide
\Concise
```

Herman Haverkort will, later on in this article, explain the new and hitherto unknown commands here. I’m glad he does. They fill me with wonder. I began creating the macros only after looking hard and intensely into the manual material… and immediately after I’d finished writing them, I stacked them away in a separate style file . That way I don’t have to see them so often and I can try to forget they’re there at all!

Now I will first show you one of the first pages, where I explain to the readers what versions there are and what version they’re holding. Next, I will try my luck at displaying the result for you…

```
\ForExclusive

\noindent{This \textsc{scenario} comes in
three different prints, stemming from the same
source file: an abridged ‘Wide’ version for
presenters and other people who are involved
with the Meridian Arts Ensemble, a more
complete ‘Circle’ version for some people who
work together with the Ensemble and a full
‘Exclusive’ version with some private details
that are only useful for communication
between the members of the Ensemble and the
Meridian Foundation.

% reserve sign for HH’s address, using macros
% defined in hhmuf.sty:
\mufhire emailhh:{herman@fgbbs.iaf.nl}

\MarkThisExclusive{This is an example of the
‘roadsign’ used in the margin of text parts
which are only visible in the ‘Exclusive’
version. It was designed for us by Herman
Haverkort\muf emailhh:{}, a grand \textsc{mae}
fan and \TeX\ wizard. Texts marked in this
manner are represented as ‘(\ldots)’ in the
‘Circle’ and ‘Wide’ printings.}

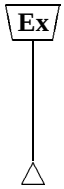
\MarkThisCircle{This is an example of the
‘roadsign’ used in the margin of text parts
which are only visible in the ‘Circle’ and
‘Exclusive’ version. It was designed for us
by Herman Haverkort\muf emailhh:{}, a grand
\textsc{mae} fan and \TeX\ wizard. Texts
marked in this manner are represented as
‘(\ldots)’ in the ‘Wide’ printing.}

Furthermore, a concise version of this
scenario consists of only the first sections,
as an introduction to presenters.

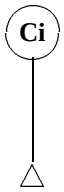
\textbf{NOTE: this is a print of the \ForWhom\
version.}}
```

which results in:

This SCENARIO comes in three different prints, stemming from the same source file: an abridged ‘Wide’ version for presenters and other people who are involved with the Meridian Arts Ensemble, a more complete ‘Circle’ version for some people who work together with the Ensemble and a full ‘Exclusive’ version with some private details that are only useful for communication between the members of the Ensemble and the Meridian Foundation.



This is an example of the ‘roadsign’ used in the margin of text parts which are only visible in the ‘Exclusive’ version. It was designed for us by Herman Haverkort,∨ a grand MAE fan and TEX wizard. Texts marked in this manner are represented as ‘(…)’ in the ‘Circle’ and ‘Wide’ printings.



This is an example of the ‘roadsign’ used in the margin of text parts which are only visible in the ‘Circle’ and ‘Exclusive’ version. It was designed for us by Herman Haverkort,∨ a grand MAE fan and TEX wizard. Texts marked in this manner are represented as ‘(…)’ in the ‘Wide’ printing.

Furthermore, a concise version of this scenario consists of only the first sections, as an introduction to presenters.

NOTE: this is a print of the ‘Exclusive’ version.

The hhparmrk kernel consists of the environments `bracespanned` and `markspanned`, which I will present now, starting with `bracespanned`.

∨ herman@fgbbs.iaf.nl

bracespanned

FG The environment `bracespanned` can be used to set paragraphs braced like this one. This paragraph is done with:

```
\begin{bracespanned}%
({\}:-{FG}(){\}}:-{HH})
  The environment

... concluded by:

  demonstrated here.
\end{bracespanned}
```

The nasty details which determine the way of bracing are all specified just after `\begin{bracespanned}`; the concluding `\end{bracespanned}` is always as straightforward as demonstrated here. HH

You might suspect that the left brace and comment ('FG' in the above example) are specified between left parentheses, while the right brace and comment are specified between right parentheses. Well, that is right. You do not have to specify both left and right stuff: you may leave one of them out, as in some of the examples below. The following paragraphs will all start with a box containing its bracing specification, that is: all that appears between `\begin{bracespanned}` and the text of the paragraph.

`){\}}:-{\muf:{Just an example}})` Instead of the comments 'FG' en 'HH' in the above example, you can of course specify whatever you want for a comment, for example a footnote. This paragraph provides an example using the `\muf` footnote macro, which is defined in the hhmuf package[⌇]. If you want to use standard footnotes, note that all that is spanned by `bracespanned` and the comments are so-called forbidden environments. To set a footnote you would have to use `\footnotemark` and `\footnotetext`; just using `\footnote` would not work. ⊕

`){(}:-{)})` This paragraph illustrates that any extendable mathematical delimiter symbols can be used instead of braces, even symbols which are pointing the 'wrong' way. Just replace the `{\{}` or `{\}}` in the example above by `{(}`, as in this example, or whatever symbol you like.

`){\}}:{65pt}{This may ...})` The `:-` in the examples above specifies the width of the spanning symbol plus comment. `:-` stands for the natural width of the symbol with comment, which usually satisfies. Another possible width specification is a colon followed by a braced dimension, like `:{65pt}`. Such a specification fixes the width of the symbol plus comment, thus enabling multi-line comments, like demonstrated here. This may be read by fiends and friends

`){]}{ExampleId}:{ex. i})` Sometimes it may be desirable to have the comments of several spanned paragraphs set all to the same width, thus leaving equal line widths for the spanned paragraphs. This can be accomplished by giving a width specification which consists of some braced identifier followed by a colon. The identifier may be chosen freely. ex. i

`){]}{ExampleId}:{ex. ii})` The previous paragraph and this one get the same width identifier (`ExampleId`) so that their comments are set to the same width: the natural width of the widest. As a result, the text bodies of both paragraphs are equally wide. However, in general you have to compile your document twice to get this result. If a second run may be necessary, the hhunits package issues a warning 'Unit values may have changed. Rerun to get them right.' ex. ii

`({\|}:-[50pt]{Gosh!}(` In the examples above paragraphs were indented on the sides to make room for the spanning symbols and comments. The amount of indention was automatically determined by the hhparmrk macros. This automatic determination can be overruled by specifying the amount of indention in a bracketed optional argument, given between the width specification and the comment. This paragraph provides an example: it is indented exactly 50pt. Specifying a 0pt indention would cause the spanning symbol and the comment to be set in the margin. Gosh!

`([:-()]:-)` TEX hackers who know when braces can be omitted are able to specify the way of spanning a paragraph quite elegantly — I think — as demonstrated by this paragraph.

⊕ Just an example

⌇ See the article about HH getting carried away, also published in this MAPS

markspanned

START The environment `markspanned` can be used to set three-part marks next to paragraphs. Such a mark consists of an upper part, a lower part, and a fill part in between. The upper and lower part have fixed size, but the fill part can be stretched so that the assembled mark spans the entire paragraph. This paragraph provides a simple example. FINISH

The above paragraph is typeset with:

```
\begin{markspanned}%
({\sc start}[\msprule]{\sc finish}{10pt}(
  The environment \envirname{markspanned} can
     :      :      :      :      :      :
  This paragraph provides a simple example.
\end{markspanned}
```

In the above example an upper part, a fill, a lower part and the mark separation are successively specified. The fill is the hhparmrk macro `\msprule`, which connects the upper and the lower part by a rule. The 10pt mark separation determines the smallest distance between the text and the three-part mark.

The nasty details which determine the way of marking are all specified just after `\begin{markspanned}`; the concluding `\end{markspanned}` is always as straightforward as demonstrated above. The following paragraphs will all start with a box containing its marking specification, that is: all that appears between `\begin{markspanned}` and the text of the paragraph.

ST `({\sc st}[\msprule]{\sc fi}[r]{10pt}(` In the example above the mark parts are centered with respect to each other. Instead of centering one can force left or right alignment by means of `[l]` or `[r]` just after the definition of the lower part. This paragraph gives an example of right alignment. FI

Until now marked paragraphs were automatically indented just enough to make room for the marks so that they did not stick out into the margins. Like with `bracespanned` one can control the amount of indention 'manually' by specifying an optional argument, just after the mark separation.

∩ `({$\cap$}{$\cup$}{5pt}[20pt](` This paragraph provides an example. It is indented exactly 20pt. This paragraph also shows that the fill part of a mark is optional and may be left out. ∪

/ `({$/$}{\bs}{5pt}(){\bs}{$/$}{5pt})` \ (`\bs` assumed to be defined as `$\backslash$`) Of course three-part marks could be set on the right by using right parentheses instead of left ones, just like with `bracespanned`. Three-part marks on both sides are possible too, like demonstrated here. \ /

More about the Fill Part

As shown in the above examples, the second argument of a three-part mark specification determines the fill part of the mark. You may omit this specification: in that case an empty fill is used. Besides `\msprule` and the empty fill one could use any desired self-made fill as long as the following is regarded:

- the fill should be a macro that takes one argument: the required size. For example `\msprule` is defined by `\newcommand\msprule[1]{\vrule height #1 width \fboxrule}`.
- the width of the fill is not taken in account when determining the positioning of the mark. Therefore the width of the fill should not be greater than both the width of the upper part and the width of the lower part of the three-part mark.

Traffic Signs

hhparmrk contains the following definition (shown here in syntactically simplified version):

```
\newenvironment{trafficsigned*}[2]{%
    \begin{markspanned}(%
    {#1{\separbox{2pt}{\large\bf #2}}}%
    [\msprule]%
    {\sepbox(0pt,1pt,0pt,0pt){%
        \large\ensuremath{\bigtriangleup}}}%
    {1em}(%
  }{%
    \ifhmode\strut\fi
    \end{markspanned}%
    \scopecorrection
  }
```

A `\begin{trafficsigned*}{\trapbox:}{A}` The environment `trafficsigned*` produces a three-part mark on the left which forces the signed text to indent. Its upper part is the second argument, boxed by the hhflxbox macro `\separbox`[1] and the tokens specified by the first argument. These are typically framing macros like `\trapbox:` (defined in hhparmrk; sets a trapezium frame), `\ringbox:` (defined in hhflxbox; sets a circle frame), or `\setlength\fboxsep{0pt}\fbox` (sets a rectangular frame). These paragraphs show some possible results. The TEX code used to start each paragraph is shown in the boxes at the beginnings. Each paragraph is ended in the source file by `\end{trafficsigned*}`. △

B `\begin{trafficsigned*}{\ringbox:}{B}` Before the `\end{markspanned}` in the definition of `trafficsigned*` a conditional `\strut` is added. This is to prevent the foot of the sign from ostensibly floating according to the depth of the last line of a signed paragraph. The `\scopecorrection` is not really needed in most cases but it guards against some rare mysterious errors. See the section about *TEXnical Details*, subsection *hhparmrk: Parallel Marks?* for explanation. △

[1] See the article about HH getting carried away, also published in this MAPS

C ... *}{\setlength\fboxsep{0pt}\fbox}{C} Besides `trafficsigned*` there exists a similar environment `trafficsigned` which sets the traffic sign in the left margin. To avoid letter-traffical collisions it is not demonstrated here.

TEXnical Details

I will not present the definition of `bracespanned` here: it is too long and complicated. There is lots of fuss in it, caused by the need or wish to parse a lot of obligatory and optional arguments which determine the exact way of bracing. However, I would like to lift a corner of the veil which covers `bracespanned`, to give hackers some idea of what is going on behind the scenes. Maybe, if I am lucky, there is some hacker out there who will be highly amazed by unnecessary complexity in my approach, and will offer me a simpler approach instead.

Fooling TEX's Gluing

My first try to build a useful `bracespanned` environment or macro consisted of straightforward use of a mathematical display. It is not difficult to set a brace spanning a box with multiple lines of text in a mathematical display environment. Alas that did not work out properly in all cases. When `bracespanned` text was surrounded immediately by normal text, the interline skip between the top spanned line and the first unspanned line above was too small, as was the distance between the bottom spanned line and the first unspanned line below. TEX considered the whole mathematical display to be one unusually high line of text. Therefore TEX did its very best to squeeze the display in at the place of one normal text line, although the display actually contained several lines.

So I decided that I had to fool TEX a bit. I constructed a mathematical display as before, but now I boxed it. Then I typeset the spanned text again behind the scenes, now using `\vtop`, to determine the height of the first line. I then shifted down the boxed display to make it have that same height. Finally I typeset the spanned text a third time behind the scenes, now using `\vbox`, to determine the depth of the last line of spanned text. Then I would insert the display, which had the height of its first line of text, and fool TEX by setting `\prevdepth` to the depth of the last line. TEX still considered the display to be a single high line, but I made TEX 'think', with respect to setting interline glue, that the display had the height of its top line and the depth of its bottom line, as if all lines in between were not there.

The approach described above had a major disadvantage: the spanned text was typeset three times. This was not only inefficient; it was also error-prone. For example: if a counter was stepped in the spanned text, then it was stepped three times. I solved this by boxing the spanned text once, using `\vtop`. Then I made a centered copy of the resulting box, a `\vphantom` of which I used to set the braces in mathematical displays. I shifted the displays down to make them have the same height as the `\vtop`ed text, so that I could put the displays and the `\vtop`ed text together.

Finally I made a copy of the `\vtop`ed text, which I unboxed to get its last line with `\lastbox` so that I could examine its depth. Then the remaining part of the procedure was like described in the previous paragraph.

All this resulted in the TEX code below (shown here abridged and simplified):

```
 %Box what has to be spanned in \@tempboxa:
\setbox\@tempboxa\vtop{#1}%
 %Make a centered copy of the result:
\sbox\@tempboxd{\ensuremath{\vcenter{%
   \copy\@tempboxa}}}%
 %Determine how much the displays should be
 %shifted down; store result in \@tempdima:
\setlength\@tempdima{\ht\@tempboxd}%
\addtolength\@tempdima{-\ht\@tempboxa}%
 %Set the left brace in \@tempboxb:
\sbox\@tempboxb{%
 \lower\@tempdima\hbox{%
 %...
 %in hhparmrk.sty one finds at this place
 %the math display stuff which sets the left
 %brace, using \vphantom{\copy\@tempboxd} to
 %determine the height
 %...}}%
 %Set the right brace in \@tempboxc:
\sbox\@tempboxc{%
 %...
 %same story
 %...}%
 %Now determine the depth of the last line:
 %Make a discardable copy of \@tempboxa in
 %\@discabox:
\setbox\@discabox\copy\@tempboxa
\setbox\@discabox\vbox{%
 %Get the last line:
 \unvbox\@discabox
 \setbox\@discabox\lastbox
 %Save its depth in \h@virtualdepth:
 \global\h@virtualdepth\dp\@discabox}%
 %Finally put it all together:
\hbox{\llap{\box\@tempboxb}%
 \box\@tempboxa\rlap{\box\@tempboxc}}%
 %Fool TeX's gluing:
\prevdepth\h@virtualdepth
```

Banishing Stubborn White Space

Some environments like to surround themselves by vertical white space. Section headings have the same tendency. But when complete (sub)sections are spanned by `bracespanned`, we do not want to get results like this:

Heading

lots of blah…

It looks ugly. Vertical space added in the beginning of a spanned passage should be squeezed out: it should be set on top of the span, instead of *in* the span. This is implemented as follows. In the beginning of a passage being boxed `\prevdepth` is set to -4774pt (just some value smaller than -1000pt. `\addvspace` is redefined to set vertical space only when `\prevdepth` is greater than -4774pt, that is: when we are not in the beginning of the

spanned passage anymore. If `\prevdepth` still equals -4774pt then the vertical space is added to a box register which holds the squeezed out space. After the complete passage has been boxed, first the squeezed out space register is unboxed and added to the main vertical list, and then the boxed passage is spanned and set. This results in the following TEX code for setting brace spans:

```
 %Box what has to be spanned in \@tempboxa:
\setbox\@tempboxa\vtop{\topsqueezeout #1}%
 %Make a centered copy etc. (see the previous
 %listing)
 %...
 %Finally put it all together:
\topsqueezein
\hbox{\llap{\box\@tempboxb}%
 \box\@tempboxa\rlap{\box\@tempboxc}}%
 %Fool TeX's gluing:
\prevdepth\h@virtualdepth
```

where `\topsqueezeout` takes care of redefining `\addvspace`, and `\topsqueezein` adds the accumulated squeezed out space:

```
\newbox\h@tsqo@squeeze

\def\topsqueezeout{%
  % Save original \addvspace:
  \let\h@tsqo@addvspace=\addvspace
  % Redefine \addvspace:
  \def\addvspace##1{%
    \ifdim\prevdepth>-4774pt\relax
      % If we are not in the beginning of the
      % box anymore, call original \addvspace:
      \h@tsqo@addvspace{##1}%
    \else
      % else add space to box register which
      % holds the squeezed out space:
      \global\setbox\h@tsqo@squeeze\vbox{%
       \unvbox\h@tsqo@squeeze
       \h@tsqo@addvspace{##1}}
    \fi}%
  % Initialize box holding squeezed out space:
  \global\setbox\h@tsqo@squeeze\vbox{}%
  % Set \prevdepth to -4774pt to indicate the
  % beginning of the box:
  \setlength\prevdepth{-4774pt}}

\def\topsqueezein{\unvbox\h@tsqo@squeeze}
```

The real implementation in hhparmrk is more complex: it also redefines `\addpenalty`, and it contains more fuss to account for nested `bracespanned` environments.

At the bottom of the spanned passage vertical space should be squeezed out as well. This is also done using a box register to hold the squeezed out space. After the passage being boxed has been entirely added to the box in which it is set, the space at the end is examined with `\lastskip`. The space is removed with `\unskip` and added to the box holding squeezed out space. Because there may be multiple skips at the end of the passage this procedure is repeated until `\lastskip` returned zero three times. I could not find a way to distinguish zero skips and no skips: `\lastskip` returns zero in both cases. Since three consecutive zero skips seem to be unlikely, the algorithm terminates when `\lastskip` yielded zero three times consecutively.

hhparmrk: Parallel Marks?

hhparmrk actually stands for: *Herman Haverkort's parallel marks*. The 'philosophy' behind this is that hhparmrk's marks should not interfere with the hierarchical structure of the document. Ideally marked and unmarked passages are typeset and processed just like they normally are, except for the presence of the marks.

In practice this is not fully attainable. First it is probably inevitable to set each marked passage as a separate paragraph, and that is what is done indeed.

A second problem is that marked passages are set in internal vertical mode, which causes footnotes and marginal notes to disappear. For hhmuf's style[l] footnotes this problem has been solved. For standard footnotes this problem can be solved, but I did not bother to do it yet.

A third problem is the grouping invoked by using environments and boxing commands. This grouping causes the scope of local assignments in marked passages to be reduced. Because that is exactly what is expected when LATEX's environments are used I decided not to do much about it, to avoid confusion. However, I built in a small 'scope correction' which suppresses the scope reduction of assignments to `\everypar`, `\par`, `\@par` and `\@currentlabel`. The first three should not be really necessary, but the handling of `\@currentlabel` can be useful when section headings or the like are spanned for some reason. The scope correction can be activated by the macro `\scopecorrection`, which is defined as follows:

```
\def\scopecorrection{%
  \h@savelocals
  \h@restorelocals}

\def\h@savelocals{%
  \global\h@sc@everypar=\everypar
  \global\let\h@sc@par=\par
  \global\let\h@sc@@par=\@par
  \global\let\h@sc@@currentlabel=
                                \@currentlabel}

\def\h@restorelocals{%
  \aftergroup\h@@restorelocals}
\def\h@@restorelocals{%
  \everypar=\h@sc@everypar
  \let\par=\h@sc@par
  \let\@par=\h@sc@@par
  \let\@currentlabel=\h@sc@@currentlabel}
```

Where to Get what Files?

To be able to use hhparmrk, you should also have the packages hhflxbox, hhunits, hhqueue and hhutils0 available. These packages are automatically loaded by hhparmrk. All files needed can be obtained from FGBBS$^{\infty}$ by requesting the file `hh.arj`. I will try to submit the packages to CTAN as well. Note that LATEX2.09 versions are not available.

---

l See the article about HH getting carried away, also published in this MAPS

∞ FGBBS — tel. (085) 217041

Sorting in TEX's Mouth*

Bernd Raichle

Stettener Str. 73
D-73732 Esslingen, FRG
raichle@Informatik.Uni-Stuttgart.de

Abstract

TEX's macro processor, the so-called *mouth*, can be used to perform very complex tasks. Because this part of TEX's programming language is as powerful as a Turing machine, it is possible to implement algorithms using only TEX's mouth.
I will show how sorting algorithms can be implemented in a straight-forward and very elegant and understandable way using only macros and macro expansion TEXniques.

1 Motivation

While reading Jeffrey's paper [2] about list processing in TEX's mouth in 1990, I figured I understood his macros and the underlying ideas — at the beginning, at least. My first attempts to implement a quicksort algorithm based on his insertion sort macros failed disastrously for a simple reason: I had a wrong model for TEX's macro processor. I thought in terms of a procedural or functional model in which a macro is seen as a *function* reading some tokens as its arguments and returning a token list as its function value, instead of thinking of it as a simple *replacement* of tokens by other tokens.

In the summer of 1993 with more experience in TEX macro programming, I retried the implementation of a quicksort algorithm with more success. Coincidentally at this time van der Laan's papers [6, 7] appeared, the first including an attempt at mouth-only processing, the second with his version of multi-purpose sorting macros.

This paper is my answer to van der Laan's questions about the usefulness and the need for mouth processing.

2 TEX's programming language

Users of TEX have different models for TEX. Depending on the user's needs, different parts of TEX and different abstraction levels of functionality are useful and appropriate. Someone who uses TEX as a text formatter via LATEX needs a different understanding than someone who uses TEX as a programming language writing macros for complicated tasks.

For a user, focussing on the programming language TEX, the language can be divided in two major parts. On one side, TEX contains the *mouth*, a macro processor providing a macro language with its own characteristics. On the other side, we have the *stomach* of TEX, the language part, where all commands with side effects, such as all assignments and text output or `dvi` output commands, are executed. The programming language realized by the stomach is incomplete because it misses control structures such as conditionals or loops.

The usable programming language has to consist of both parts because the assignment capabilities of the stomach are needed in order to define macro definitions and to read or write text, i.e., to produce an output. The mouth is necessary for all tasks that needs an iterative application or a recombination of input tokens. TEX's stomach uses the macro processor for almost all commands to scan the command arguments. Additionally, while scanning the arguments of many stomach commands, such as `\write`, `\edef`, count or dimension register assignments, all tokens are expanded. Thus stomach operations are not allowed in these places, leading to the problem of *fragile* commands in *moving* arguments. This is partly taken care of with LATEX's `\protect`.

Because of the importance of TEX's mouth, the rest of this paper focusses on the macro processor part in more detail.

2.1 TEX's macro processor

TEX's mouth realizes a macro language operating on token lists.[1] A token references either a primitive command or a macro definition and is built from the characters of the input files in TEX's *eyes* and *mouth* using a fixed set of rules [5, pp. 37ff].

The mouth reads one token after another from the currently active input, a file or a token list, and tries to expand each token. If the token read is unexpandable, i.e., it references a primitive with side effects, the expansion process is stopped

*Reprint from the Proceedings of the Eigth European TEX Conference, Gdańsk, Poland, September 26–30, 1994.

[1] Be aware that this statement and the use of the word *mouth* in this paper is imprecise as a result of Knuth's 'technique of deliberate lying'. More precisely, TEX's *gullet* is the macro processor, whereas the *mouth* is 'the process by which input files are converted to lists of tokens' [5, p. 267].

and the token is given to the stomach for further processing. If the token is expandable, i.e., it references a macro or a primitive without side effects, the mouth reads more tokens *without* expansion, if arguments are needed, and replaces the token read in the input by a list of tokens. This means that the next token is the first token of the replacement text because the original token is removed.

Comparing macros and macro expansion with procedures or functions in a procedural programming language, it is important to note that tokens building the arguments of a macro are *not* expanded at the time the macro is expanded.

2.2 Basic Operations

TEX's macro language only knows about a very restrictive set of basic operations on (token) lists implementable by very simple macro definitions. These basic operations[2] are

- get first element of a list `\def\Car#1#2\EOL{#1}`,
- get the rest of a list `\def\Cdr#1#2\EOL{#2\EOL}`,
- add a new element in front of a list `\def\Cons#1#2\EOL{{#1}#2\EOL}`,
- add a new element to the end of a list `\def\tCons#1#2\EOL{#2{#1}\EOL}`,
- append two lists `\def\Append#1\EOL#2\EOL{#1#2\EOL}`,
- and similar operations to get the n-th element of a list, add n elements, or append n lists to a list with $n < 9$.

Complex operations, such as

- get the length of a list,
- search for an element in a list,
- return the n-th element of a list,
- delete all elements equal to a specified element,
- reverse a list,
- apply an operation to all elements of a list,

and a lot more operations are not supported by simple macros, but have to be defined using more or less complex macro definitions. When looking on the list of complex operations, it should be obvious that they are based on an important concept: *iteration!*

2.3 Loops and Iteration

The macro language does not contain any loop primitives or any other primitives allowing iteration. The reason is simple: it is not necessary. In a macro language missing an iteration primitive, a loop can be easily implemented by using the macro token realizing the loop in its own replacement text. That is, iteration is implemented by using recursion.

If the replacement text of a macro contains a token referencing this macro, the expansion process in the mouth will expand the macro and expand the macro and expand the macro . . . .

Example 1 We want to apply a macro `\Func` to all tokens of a list. A first definition for the iteration macro is

```
\def\Mapc#1{\Func{#1}\Mapc}
\def\Func#1{ (#1) }
```

Applying this macro to the token list `1234`, will yield the expansion sequence

```
\Mapc                    1 2 3 4
\Func{1}\Mapc              2 3 4
   (1)  \Mapc              2 3 4
   (1)  \Func{2}\Mapc        3 4
   (1)     (2)  \Mapc        3 4
   (1)     (2)  \Func{3}\Mapc 4
   (1)     (2)     (3)  \Mapc 4
  . . .
```

This first macro definition seems to solve our problem. But what will the macro `\Mapc` do after reading the last element `4`? It will continue and will finally stop with an error message since we have implemented an endless loop.

Before solving this problem, a small change to the macro definition will show another iteration macro technique which can be used to collect temporary values.

Example 2 We want to collect the application of a macro `\Func` to each element in a list without applying this macro, until we have iterated over the complete list.

```
\def\MapCar{\DoMapCar{}}
\def\DoMapCar#1#2{\DoMapCar{#1\Func{#2}}}
```

Before the iteration starts, the value of this argument, in which we collect the result, is initialised with the empty list. Applying the new macro to the list `1234` yields the sequence

```
\MapCar                          1 2 3 4
\DoMapCar{}                      1 2 3 4
\DoMapCar{\Func{1}}                2 3 4
\DoMapCar{\Func{1}\Func{2}}          3 4
\DoMapCar{\Func{1}\Func{2}\Func{3}} 4
  . . .
```

An advantage of `\MapCar` in comparison to `\Mapc` is important: because the collected result is an argument of `\DoMapCar`, it is possible to enhance this macro by applying additional operations after the iteration is completed.

To make the two macros perfect for common use, it is necessary to add tests checking for the end of the list which terminates the recursion.

Example 3 The definitions of the two macros `\Mapc` and `\MapCar` are completed by adding `\if...` comparisons checking for the end of the argument token list. I will use the token `\relax` in all following macro examples as the end of list marker. Be aware of this in case you want to use these macros with lists containing `\relax` as a normal element.

[2] The lists in the examples are token sequences delimited by the token `\EOL`.

```
\def\Mapc#1{\DoMapc#1\relax}
\def\DoMapc#1{%
  \ifx\relax#1%  end of list?
  \else
    \ReturnFi{\Func{#1}\DoMapc}%
  \fi}

\def\MapCar#1{\DoMapCar{}#1\relax}
\def\DoMapCar#1#2{%
  \ifx\relax#2%  end of list?
    \ReturnElseFi{#1}%
  \else
    \ReturnFi{\DoMapCar{#1\Func{#2}}}
  \fi}

\def\ReturnFi#1\fi{\fi #1}
\def\ReturnElseFi#1\else#2\fi{\fi #1}

\def\Func#1{(#1)}% ... as an example
```

When adding `\if...` comparisons to complete these two macros, it is necessary to ignore all tokens of the false branch of the test including the `\else` and `\fi` tokens. Otherwise, the following iterations will use these tokens instead of the next element of the list as its arguments. Additionally, the definition of a macro without ignoring these tokens applied to a list will overflow some of TEX's internal stacks.

To avoid this problem, I have used the special macros `\Return...` in the `\Mapc` and `\MapCar` macros. In most cases `\expandafter` can be used instead of the `\Return...` macros to skip over the `\else` and `\fi` [3], but sometimes a 'slightly ridiculous sequence' of `\expandafter`s is needed [1].

3 Sorting

Sorting is a basic tool whose algorithms and algorithm implementations are complex and interesting enough to use it for studying the advantages and disadvantages of a programming language. In the rest of this paper, I will explain the implementation of sorting algorithms in TEX's macro processor by using the technique shown in the `\MapCar` macro.

From the set of well-known internal sorting algorithms [4, pp. 73ff], such as sorting by insertion (straight insertion sort, Shell's sort), by selection (straight selection sort, Heapsort), by exchanging (bubble sort, shaker sort, Quicksort), and by merging (merge sort), I will describe and implement Quicksort in detail because this sorting method is appropriate and fast enough for larger lists.

3.1 Quicksort

The principle of the quicksort method is easily explained: Given is a field L with n elements. Choose a key value K. Partition the field L in two subfields L_l and L_r such that $\forall x \in L_l : x \leq K$ and $\forall x \in L_r : x \geq K$. Apply these partitioning steps to the subfields recursively until all subfields contain at most one element. The concatenation of all subfields is the sorted field.

A difficulty in implementing the quicksort method is the selection of the key value K at the beginning of each partitioning step. If K can be chosen in such a way that the two resulting subfields contain an equal number of elements, the execution time of the algorithm will be of order $n \log n$. In the worst case, one of the two subfields contains only one element, so that the execution time will be of order n^2. Because the search for the best value K in each subfield needs additional execution time, simpler selection methods are usually implemented: choose a random integer between the value of the first and the last element or consider a small sample and choose the median of this sample.

When implementing the quicksort method using arrays, the partitioning needs to be executed 'in place' because of memory restrictions. In those implementations elements are exchanged in pairs until the field is partitioned into the two subfields [4, pp. 114ff].

3.2 Quicksort and Lists

When trying to implement the quicksort method using TEX's macro processor, we have to use the description above for (token) lists. Ignoring the problem of selecting a good key value K, we can use the following algorithm:

Given is a list with more than one element. Choose the first element as the key value K; the two sublists L_l and L_r are empty. Compare each element x in the rest of the list with the key value K. If $x \leq K$, append it to L_l, otherwise to L_r.

Example 4 Given is a list $L = \{3, 6, 1, 8, 7, 2, 5, 0, 4\}$. We choose the first element (3) as key value and initialise the two sublists with $L_l = \{\}$ and $L_r = \{\}$. After iterating over all elements remaining in the list, the two sublists are $L_l = \{1, 2, 0\}$ and $L_r = \{6, 8, 7, 5, 4\}$.

If this partitioning step is applied recursively to all sublists, and at the end these sublists and key values are concatenated in the correct order, we get a sorted list.

An important fact is that a simple iteration is the only complex list operation in this algorithm, otherwise only primitive list operations (get first element, get the rest of a list, append lists) are used.

3.3 Quicksort Implementation

Using the technique of the `\MapCar` macro, an implementation of this algorithm is easy:

```
\def\QuickSort#1{\StartPartition#1\relax}
\def\StartPartition#1{%
  \ifx\relax#1%    % empty list?
  \else \ReturnFi{\DoPartition{#1}{}{}}%
  \fi}

\def\DoPartition#1#2#3#4{%
  \ifx\relax#4%    % end of rest?
    \ReturnElseifFiFi{\QuickSort{#2}%
                        {#1}%
                      \QuickSort{#3}}%
  \else\ifnum#4<#1 % element < key value?
    \ReturnElseFiFi
          {\DoPartition{#1}{#2{#4}}{#3}}%
  \else \ReturnFiFi
          {\DoPartition{#1}{#2}{#3{#4}}}%
  \fi\fi}
```

```
\def\ReturnFi#1\fi      {\fi #1}
\def\ReturnElseifFiFi#1\else#2\fi\fi
                        {\fi #1}
\def\ReturnFiFi#1\fi\fi {\fi\fi #1}
\def\ReturnElseFiFi#1\else#2\fi\fi
                        {\fi\fi #1}
```

When the macro `\QuickSort` is applied to the list `{{3}{6}{1}{8}{7}{2}{5}{0}{4}}` of Example 4 and `\tracingmacros` is set to a non-zero positive number, the following sequence can be observed in the lines of the `log` file where the `\QuickSort` macro is called:

```
{3}{6}{1}{8}{7}{2}{5}{0}{4}
{1}{2}{0}
{0}
{2}
{6}{8}{7}{5}{4}
{5}{4}
{4}
{8}{7}
{7}
```

This sequence is the result of the partitioning tree shown in Figure 1. Looking at the leaves of this tree, you will see the sorted list.

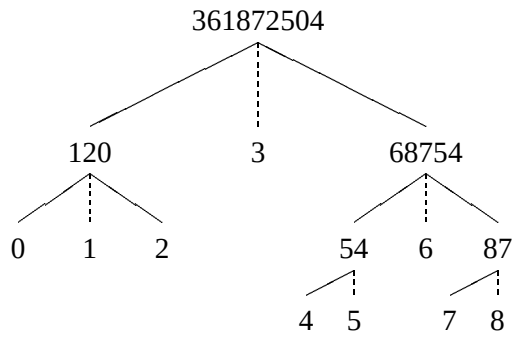


Figure 1: *Sublist partitions (solid lines) and the used key values K (dotted lines) for the example* `\QuickSort{361872504}`.

3.4 Necessary Improvements

Because the first implementation of quicksort uses a very simple scheme to choose the key values K, it is normal that tests will show worst cases of the execution time when these macros are applied to already sorted lists or to lists where all elements are equal.

40 elements	(a)	(b)	(c)	(d)
time	18	41	stack overflow	36

Table 1: *Execution time (in seconds) of the simple Quicksort implementation for a list with 40 elements.*

I have applied the first quicksort implementation to a list with 40 elements (a) in random order, (b) sorted in ascending order, (c) sorted in descending order, and (d) with equal elements. Table 1 shows the results obtained on a slow personal computer (Atari ST, 68 000/8 MHz).

Case (c), the sorted list in reverse order, is the worst case for this implementation: if the list contains more than some 30 elements, TEX will abort with an overflow of the parameter stack.

To overcome this problem, it is necessary to either use a better selection scheme for the key values, which is impossible without using more complex operations, or to ensure that the list and sublists are not sorted for *all* partitioning steps.

Example 5 If each odd numbered element of the list $A = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ is appended to the tail and each even numbered element to the front of the list $B_0 = \{\}$, the result is the sequence of lists $B_1 = \{0\}$, $B_2 = \{1, 0\}$, $B_3 = \{1, 0, 2\}, \ldots, B_9 = \{7, 5, 3, 1, 0, 2, 4, 6, 8\}$.

A very simple method to disturb the order in a list is sketched in Example 5. Applying this method to the implementation when appending an element to one of the two sublists, a better behaviour for the worst cases is attained.

```
\def\QuickSort#1{\StartPartition#1\relax}
\def\StartPartition#1{%
  \ifx\relax#1%    % empty list?
  \else \ReturnFi{\DoPartitionI{#1}{}{}}%
  \fi}

\def\DoPartitionI#1#2#3#4{%
  \ifx\relax#4%    % end of rest?
    \ReturnElseifFiFi{\QuickSort{#2}%
                          {#1}%
                       \QuickSort{#3}}%
  \else\ifnum#4<#1 % element < key value?
    \ReturnElseFiFi
         {\DoPartitionII{#1}{#2{#4}}{#3}}%
  \else \ReturnFiFi
         {\DoPartitionII{#1}{#2}{#3{#4}}}%
  \fi\fi}

\def\DoPartitionII#1#2#3#4{%
  \ifx\relax#4%    % end of rest?
    \ReturnElseifFiFi{\QuickSort{#2}%
                          {#1}%
                       \QuickSort{#3}}%
  \else\ifnum#4>#1 % element > key value?
    \ReturnElseFiFi
         {\DoPartitionI{#1}{#2}{{#4}#3}}%
  \else \ReturnFiFi
         {\DoPartitionI{#1}{{#4}#2}{#3}}%
  \fi\fi}
```

A drawback of this disordering trick should be noted: whereas the first quicksort implementation is *stable*, i.e., elements with equal keys retain their original relative order, the new one is unstable.

3.5 Comparison with other Quicksort Implementations

In order to compare this fully expandable version of quicksort with other quicksort versions in TEX, I have used two other, not fully expandable implementations: line (I) in Table 2 represents the execution times for the shown `\QuickSort` macro, line (III) shows the result of Kees van der Laan's multi-purpose implementation [7]. Finally, I tried to implement an optimized quicksort version according to Knuth's description [4, S. 114ff], using a median of three values for the key values K in each partitioning step. Line (IV) shows the result for this version.

# elements	1	20	40	80	160	320
(I)	11	14	20	46	90	289
(II)	11	13	18	36	66	196
(III)	14	17	26	34	68	162
(IV)	10	13	16	25	45	93

Table 2: *Execution time (in seconds) of the four Quicksort implementations for disordered lists.*

Whereas the execution times of the two quicksort implementations (III) and (IV) are of order $n \log n$, the execution time of the 'mouth only' implementation (I) is of order n^2. The implementation uses the quicksort method — why is the execution time not proportional to $n \log n$? The answer to this question is surprising: In the partitioning loop, TEX has to scan the tokens of the two collected sublists as arguments of the `\DoPartition` macro and it has to skip these tokens in the false branches of the used conditions using the `\Return...` macros. Nonetheless the table shows that the presented `\Quicksort` macro is fast enough for short lists with less than 50–60 elements.

Using this analysis of the problem, it is possible to speed up the `\QuickSort` macros: We move the arguments for the recursive calls in `\DoPartitionI/II` to the end and replace the calls by new macros with the same argument structure. These new macros expand to the former macro sequences in the conditional branches with the necessary argument recombination. With these changes, TEX has to scan the sublists in `#2` and `#3` only twice instead of three times. The macros of the final version of `\QuickSort` are shown in the next section; line (II) of Table 2 contains the measured times.

3.6 Implementation Enhancements

The macros shown implement a quicksort algorithm suitable only for integers. After substituting the integer comparison tests `\ifnum#4<#1` (and `\ifnum#4>#1`) by a `\Compare` macro and using an appropriate definition for this macro, these macros can be used to sort other data types.

```
\def\QuickSort#1{\StartPartition#1\relax}
\def\StartPartition#1{%
  \ifx\relax#1%    % empty list?
  \else \ReturnFi{\DoPartitionI{#1}{}{}}%
  \fi}
\def\DoPartitionI#1#2#3#4{%
  \ifx\relax#4%    % end of rest?
    \ReturnElseifFiFi\DoQuickSort
  \else\Compare{#1}{#4}%
    \ReturnElseFiFi\PartitionGreaterII
  \else
    \ReturnFiFi\PartitionLessII
  \fi\fi
  {#1}{#2}{#3}{#4}}
\def\DoPartitionII#1#2#3#4{%
  \ifx\relax#4%    % end of rest?
    \ReturnElseifFiFi\DoQuickSort
  \else\Compare{#4}{#1}%
    \ReturnElseFiFi\PartitionLessI
  \else
    \ReturnFiFi\PartitionGreaterI
  \fi\fi
  {#1}{#2}{#3}{#4}}
\def\DoQuickSort#1#2#3#4{%
  \QuickSort{#2}\Func{#1}\QuickSort{#3}}
\def\PartitionLessI    #1#2#3#4{%
  \DoPartitionI{#1}{#2}{{#4}#3}}
\def\PartitionLessII   #1#2#3#4{%
  \DoPartitionII{#1}{#2}{#3{#4}}}
\def\PartitionGreaterI #1#2#3#4{%
  \DoPartitionI{#1}{{#4}#2}{#3}}
\def\PartitionGreaterII#1#2#3#4{%
  \DoPartitionII{#1}{#2{#4}}{#3}}


\def\Func#1{{#1}}
\def\Compare#1#2{\ifnum #1>#2\space}
```

The `\QuickSort` macros do not depend on the macro `\Compare` being fully expandable. However, only if `\Compare` is fully expandable will the resulting `\QuickSort` be fully expandable.

In the following example for a `\Compare` macro, a list of text words is sorted by the length of the typeset text using a roman font.

```
\def\Compare#1#2{\begingroup \rm
  \setbox0=\hbox{#1}%
  \setbox2=\hbox{#2}%
  \expandafter\endgroup\ifdim\wd0>\wd2 }
\def\Func#1{#1\par}
```

In order to allow the postprocessing of the sorted list, each element is handed to the macro `\Func`, which can be changed appropriately.

4 Conclusions

Is it necessary to restrict the implementation of the sorting algorithm to mouth processing? Or using van der Laan's words [6, p. 315]: 'Why don't [the authors] make clear the *need* for mouth processing, or should I say mouth optimizing?'

Generally, it is always necessary to *use* mouth processing because the mouth is the only language part of TEX processing macros with conditionals, and hence the necessary capabilities to provide iterations.

Additionally, it is necessary to *restrict* macro definitions to mouth processing, if the context of the macro usage forces it because TEX's stomach processes the macro in *expansion only mode*, e.g. in the argument of a `\write` or `\edef` command. There are even more examples of contexts in which only mouth processing is allowed: 1) Integers, dimensions or glue specifications can be built using macros specifying different parts. If one of these macros expands to an unexpandable token, e.g. `\relax` or another stomach command, TEX stops the scanning process and probably complains about a missing number. 2) `german.sty` [8] uses an active double quote character to allow the shortcut input `"a` for an ä. Because a double quote is also used in TEX to input numbers in hexadecimal notation, it is necessary to use mouth-only processing to decide whether the double quote introduces a hexadecimal number or not. 3) Inside the mouth command `\csname...\endcsname` all used control sequences have to be expandable.

At first glance, the restriction to mouth processing seems to be necessary only for expert TEX users writing some high-level macros using low-level macros and primitives. Thinking of a LATEX beginner who needs to insert `\protect` in some obscure places, it seems that the complex interactions between TEX's eyes, mouth, and stomach cannot be hidden from ordinary users.

Finally, in order to use the full power of the programming language TEX, it is necessary to get a better understanding of the capabilities and restrictions of its parts. Furthermore, since TEX's stomach will never be used without its mouth, it is best to start playing with TEX's macro processor.

5 Acknowledgements

I am pleased to thank Alan Jeffrey for his helpful comments on a preliminary version of this paper. In particular, his idea of introducing additional macros based on my analysis of the `\QuickSort` version (I) has led to the improved version (II). Thanks to Frank Tränkle and Jörg Heitkötter (–joke) for proofreading.

References

[1] Victor Eijkhout, Oral TEX: Erratum, *TUGboat*, **13**(1):75, 1992.

[2] Alan Jeffrey, Lists in TEX's Mouth, *TUGboat*, **11**(2):237–245, 1990.

[3] Alois Kabelschacht, `\expandafter` vs. `\let` and `\def` in Conditionals and a Generalization of plain's `\loop`, *TUGboat*, **8**(2):184–185, 1987.

[4] Donald E. Knuth, *The Art of Computer Programming*, Volume 3: Sorting and Searching, Addison-Wesley, Reading, Mass., 1973.

[5] Donald E. Knuth, *The TEXbook*, Addison-Wesley, Reading, Mass., 1986.

[6] Kees van der Laan, Syntactic Sugar, *TUGboat*, **14**(3):310–318, 1993.

[7] Kees van der Laan, Sorting within TEX, *TUGboat*, **14**(3):319–328, 1993.

[8] Hubert Partl, German TEX, *TUGboat*, **9**(1):70–72, 1988.

One by one the guests arrive

Kees van der Laan

Hunzeweg 57,
9893 PB Garnwerd, The Netherlands
cgl@rc.service.rug.nl

Abstract

A plea is made for writing macros in plain TEX sufficiently documented to be used with all flavours of TEX.

1 Introduction

This note emerged form a request of Sebastian Rahtz triggered by my message which I passed along with my public appraisal for the 100 (LA)TEX FAQs.

This plea, this shout, hopes to awake the notion that we are all better off if we write macro *software* in the lowest common set of all TEX flavours. At least it might initiate a discussion because I'm realistic enough that not all involved share my views.

Of course I know that reality is more complicated, and that a right balance is the best we can opt for, so let us go for that.

2 Why

would a LATEX devotee ask? Do you have concrete arguments? Well, from my own experience I can say that there was a time I needed typesetting number ranges. Only the LATEX style of Donald Arsenau was available. But, what I needed was a few macros to cooperate with plain, so I had to write one of my own,[1] which by the way emerged as a much, much more compact suite. After all the need has faded away because I tackled the handling of bibliography references more fundamental. The point is that it would have been better if there had been a kernel independent from the higher layer which I could have taken over. The interface towards the higher level, or let us say the user interface, should better be built on top. The paradigm in this example is the awareness of CISO, as analogy of FIFO, meaning Collective In and Smallest Out, which solves the problem.

That this approach is beneficial in software engineering in general has been proven by the various numerical software program libraries, which have the basic material written in the lowest language feasible, FORTRAN, allowing stability, optimization of the code, and confidence in use. Similar, I remember the PDE (partial differential equation) packages which use common basic algorithms, but differ in the jargon at the user level. I hope that the macro/package/module writers have a feeling for the savings of the costs which can be gained over time, by this attitude. As a volunteer organization one could shrug it off and say I don't care, costs are not relevant. Then there is still another nasty guy lurking around the corner that the (All)TEX community like various sects will fall apart, will fragment. To continue the tune

```
And no one knows where the night is going
And no one knows why the wine is flowing
O love, I need you, I need you, I need you
                I need you now
```

Another example is how to provide for headings? The answer is that I don't care so much about heading macros because the common part is so negligible, while it is highly intertwined with the user interface. But—there is always a but—I for one am strongly in favour of starting from two-part macros, which should perform the essential functionalities whatever you may wish, and build all the ornantia—i.e., the user-interfaces, possibly with less functionality—on top. This approach obeys the *separations of concerns* principle, and pays off in maintance, if not that it spreads more easily.[2] To give you an idea of how I did it basically in blue.tex

```
\def\beginghead{<the required functionality>}
\def\endhead{<the required finishing off>}
%with as one-part on top
\def\head#{\bgroup\beginhead
  \aftergroup\endhead
  \afterassignment\ignorewhitespace
  \let\dummy= }
%or the tribute to manmac
\def\bluehead#1\par{\beginhead#1\endhead}
```

The last tribute lost the processing on-the-fly functionality, but most of the time I don't need that, at the expense of simpler markup. But the latter is a matter of taste, I know.

If people like a LATEX-flavoured header just go ahead and add it. The fundamental functionalities have been provided already, just a user interface has to be provided as variant.

3 Conclusion

The point I'm trying to make is that we are all better off when complex fundamental parts will be programmed in plain, perhaps after it has proven to be a fundamental point. To end Cohen's song

```
The guests are coming through
    The open-hearted many
  The broken-hearted few
```

[1] Who cares? It is estimated that 80% or more of the software is continuously rewritten, that is a fact. My reply is that we can do much better, and we should if we opt for the best.

[2] Forgive me this joke, with LATEX widespread.

BLUe's Format Databases

Stay organized

Kees van der Laan

Hunzeweg 57,
9893 PB Garnwerd, The Netherlands
cgl@rc.service.rug.nl

Abstract

The backgrounds, use, and coding of BLUe's Format Databases have been discussed. Two kinds of databases have been introduced class I (data, such as addresses, references, and script parts for example pictures), and class II (macros, such as variant formats and tools).
At the heart lies the selective loading TEXnique, which allows that only what is needed will be loaded on-the-fly. The data structures and operations on them have been treated. The use within the context of typesetting scripts have been elucidated via examples. At the end the coding is explained.

Keywords: Active list separators, addresses, comment blocks, compatible extension, data integrity, databases, education, fifo, lazy evaluation, list element tag, macro writing, mail-merge, no-nonsense, number ranges, pattern matching, pictures, plain TEX, references, reusable software parts, search, selective loading, set macros, software engineering, table of contents, variant document parts.

1 Introduction

Why couple the buzzword database to TEX? What has TEX got to do with it? Vaguely the answer is that we like to store addresses, references, copy parts, tools and formats outside of TEX—or one of its flavours—and only borrow what we need. Important is also the data integrity aspects which can be achieved via databases. We only store the information once for use in different contexts. At the heart is the process of selective loading.

Let us go back to Knuth and survey how he coped with using macros (formats and tools), addresses, references, and copy parts.

With respect to formats Knuth provided and discussed: concert format, letter format, and *The TEXbook* format, i.e., the manmac collection of macros. How did he store those macros? The answer is: simply in separate files. Is that good enough? The answer is: no, IMHO, with all respect, assumed we only need part of it. Similarly, he used gkpmac, as an independent collection.

How to handle macros has been treated in *The TEXbook* Appendix D *4. Selective loading of macros.* For references he mentioned at the end of that section '. . . to prepare a bibliography for a paper, by reading a suitably arranged bibliography file; only the entries that correspond to defined control sequences will be loaded.' No further details did he supply for how he coped with a database of addresses, or (prefab) parts. However, in manmac Knuth elaborated on how to handle the document parts

- answers, and
- index reminders, IRs for short.

Although the answers and IRs are very loosely connected to the database idea I'll survey Knuth's approach nevertheless, because they give insight how to cope with copy parts, especially how to create such files and how to process them efficiently. This elucidates the use of Knuth's \\, the what I call list element tag.

1.1 Answer elements

in the file answers obey the syntax

```
<answerelement> is
   \ansno <chapnumber>.<exnumber>
   <answerproper>
```

Example *(An answer, what is the question :-))*

```
\ansno 3.16:
The Bible Illuminated
```

The file answers contains separator lines which begin with \ansno—the list element tag—and contain the separators period and colon. Very close to natural markup. Only \ansno is extra. Apparently we can't do without something like that.

In an abstract sense the collection of answers forms a list. Lists in TEX are processed via some sort of active list separator, as explained in Appendix D of *The TEXbook*. How did Knuth cope with the list of answers[1]? For this case he used \ansno as active list 'separator,' i.e., \ansno processes each answer. Perhaps we should call this tag the *list element tag*, because it marks (the start of) a list element.

[1] Let us assume for simplicity that the answers are already stored in the file answers.

The name active list separator is not accurate enough, although conceptually it comes close. It is just the first element which is inconsistent, the element is preceded by the tag. That tag separates partially, let us say.

1.2 Index reminders

in the file index obey the syntax `<IR> is`

```
<indexelement> !<classificationnumber>
    <pagenumber>.
```

Knuth did not process this file within TEX. Ultimately, he set the sorted and enriched index file via TEX, of course.[2]

1.3 And what about addresses?

A letter is provided with the address included, no use of databases, see *The TEXbook* 404, 405.

1.4 Knuth's `\ifoption`

This is about optional inclusion of macros, and in general document parts, as mentioned in the infamous Appendix D of *The TEXbook*. Assume that a file macs has been created with as structure a sequence of `\ifoption`s.

```
\ifoption A <Apart>\fi
\ifoption B <Bpart>\fi
\ifoption C <Cpart>\fi
%et cetera
```

Knuth's selective loading comes down to

- the user supplies for example `\load{macs}{AC}`
- create the list `\options` with for example as replacement text `\\A\\C`, with `\\` the function to process the next token as argument (done by `\fifo`...`\ofif` below)
- define `\ifoption`, and therein `\\`, such that the appropriate parts of the file, macs let us say, will be loaded.

This is achieved by the following from *The TEXbook* 384, transcribed into my fifo notation

```
\def\load#1#2{%#1 file, #2 options
%Purpose: #2 elaborated into list of options
%          as replacement text of \options.
   \let\options\empty\let\\\relax\fifo#2\ofif
   \input#1 }
\def\fifo#1{\ifx#1\ofif\ofif\fi
   \process#1\fifo}        \def\ofif#1\fifo{\fi}
\def\process#1{\edef\options{\options\\#1}}
%
\def\ifoption#1{%Purpose: to locate #1
%and to load the corresponding part.
  \def\\##1{\if##1#1\resulttrue\fi}%
  \resultfalse \options \ifresult}
%with driver, for example
\load{macs}{AC}
%and the file macs with for example
\ifoption A The A part\fi
\ifoption B The B part\fi
\ifoption C The C part\fi
\endinput
```

My approach differs in that I don't have two list element tags, such as `\ifoption` and `\\`, but only `\lst`, respectively `\tool`. My A, B etc. are names, control sequences, which can be initialized with an error message, with the advantage that this message will appear when the name does not match the name of the document element to be loaded. Furthermore, I used a token variable `\namelst` instead of the definition `\options`, to store the list of options, but that is not essential.

My begin and end of the optional part are braces to facilitate using it as a replacement text. (With tools and formats I don't have braces but terminate via `\endinput`.) Knuth's begin and end tags are `\ifoption` and `\fi`.

I won't detail further but will explain the superior method of Díaz.

1.5 Díaz fast selective loading

The Díaz process, mentioned in *The TEXbook* 384 as a superior method, reads essentially as follows.[3]

```
%The TeXbook, Appendix D 4.Selective loading.
%The Max Diaz fast selective loading process.
%(A little simplified, and combined with
% my list element tag, \lst.)
\def\lst#1{\catcode`\~=
   \ifx#1\undefined14    %comment
              \else9 \fi}%ignore char
%We want to load the cgl part.
\def\cgl{<anything>}

\lst\name
~a
~b
~c
\lst\cgl
~aa
~bb
~cc
\lst\erik
~aaa
~bbb
~ccc
\catcode`\~13
%
<Commonpart>
\bye
%Explanation: the list element tag toggles the
%catcode for ~ such that either the first
%character is ignored (and the rest of the line
%loaded) or the line is a comment line.
```

In the above one can replace `\lst\name...` by `\input <filetobeloadedfrom>`.

I did not elaborate much on Max Díaz approach via catcode changes, because I don't like as yet to insert alien first line characters in the document parts stored in the file to be selectively loaded from.[4]

[2] How to handle (modest) indexes completely within TEX has been treated in 'BLUe's Index.'

[3] Courtesy Erik van Eynde for posing the practical problem of variant document parts on the TEX-nl discussion list, which stimulated me to look closer to Knuth's and Diaz' approach.

[4] I do realize the gains in efficiency though, and more importantly there are less restrictions. Perhaps I'll use it in the next release of BLUe's Format System.

Loosely related to the above is the use of the so-called block comments, to treat parts of documents as comments. What I have seen and worked on myself so far is similar to verbatims except that it is not set. I rate Diaz approach as superior and won't elaborate on block comment further.

1.6 Notations and definitions

gkpmac stands for the macros used by Graham, Knuth, Patashnik to format Concrete Mathematics.

manmac denotes the macros used by Knuth to format *The TEXbook*, and the METAFONTbook.

Pascal denotes Wirth's programming language.

When I load from a database file the extension is already provided for and reads '.dat'.

2 Databases

With algorithms,[5] and similar with databases, we have two main aspects

- data structures, and
- operations on the data.

Up till now I have used the following two kinds of databases to cooperate smoothly with TEX.

I data, such as, addresses, references, and script parts
II tools, such as, variant formats, and various macros.

I call these class I and class II databases. Samelson's principle has been obeyed

'Don't pay for what you don't use,'

which comes down to selective loading.

The difference between class I and class II is that in class II only one element has to be loaded per scan.

Each file consists of a list of `<listelement>`s. The file names obey the syntax `<subjectindicator>`.dat. I consider it useful to attach a name to each list element.

Below I'll discuss the elements proper, deprived from the useful extras such as author information, markup for selective line numbering, contents, and history of changes, in order to convey the main idea.

2.1 Class I data structure

At the moment this is used in the files

- address.dat, for addresses
- lit.dat, for references, and
- pic.dat, for pictures.

Each list element is a triple and reads as follows.

- list element tag, `\lst`
- name of list element
- the list element proper, as group. that is enclosed by curly braces.

I chose `\lst` as the tag for list element tag. One could equally well have chosen another control symbol or sequence as list element tag, for example Knuth's `\\`. However, because `\\` is so heavily used for e-o-l I refrained from that.

Within the replacement text I adopted application specific conventions. For example with addresses I used `\\`—another lower level list element tag :-)—to facilitate formatting. I also used elements tagged by `\email`, and `\phone`. When the entry is used for an address label the latter tags are `\let`-equal to `\gobble`, i.e., they are neglected for the label.

For addresses the syntax of an entry reads

```
\lst\<name>{<salutename>
\\<fullname>
\\<affiliation>
\email{...}
\phone{...}
\fax{...}
}
```

Example *(Entry for address.dat)*

```
\lst\ntg{NTG
\\Nederlandstalige \TeX{} Gebruikersgroep
\\Postbus 394
\\1740 AJ Schagen
\\The Netherlands
\email{ntg@nic.surfnet.nl}
}
```

For references the replacement text starts with the name of the author with as few punctuation marks as possible, followed by the date, title, and information about the source. I also decided to include `\annotation`, which can gobble its argument when not needed, or anything you want. In the simplest case it can return 'its argument' via `\let\annotation\relax`.

Example *(Entry for references.dat)*

```
\lst\knuthded{Knuth D.E (1984):
   Computers and Typesetting.
   \TB. Addison-Wesley.
   ISBN 0-201-13447-0 (hard cover)
   ISBN 0-201-13448-9 (soft cover).
   \annotation{For the correct printing
    look in the index for \cs{language}.
    \TeX, is frozen in version $\pi$,
    3.1415\dots}
}
```

For entries of the picture database see 'BLUe's Graphs,' the ideas are similar.

The restrictions

are that as part of the replacement text tags are excluded which are not allowed in an argument of a `\def`.

2.2 Class II data structure

At the moment this is used in the files

- fmt.dat, for variant formats, and

[5] Courtesy Niklaus Wirth 'Algorithms = data + programs.'

- tools.dat, for tools, such as the index macros.

To facilitate handling I consider it useful to close each element by `\endinput`. As a result each list element is a quadruple

- list element tag, `\tool`
- name of list element
- the list element proper
- `\endinput`.

I chose `\tool` as the tag for list element tag. One could equally well have chosen another control symbol or sequence as list element tag, for example Knuth's `\\`. However, because `\\` is so heavily used for e-o-l I refrained from that.

The list element tag name ends either with tool or fmt. Within the replacemenet text no conventions are prescribed. The syntax for an entry reads `\tool\<name>tool`

```
\tool\<name>fmt
   <toolmacros>    or        <formatmacros>
\endinput                \endinput
```

Example *(Entry for tools.dat)*

```
\tool\englishtool
\message{ ---English , Jan 95, cgl--- }
\abstractname{Abstract}
\acknowledgementsname{Acknowledgements}
\contentsname{Contents}
\indexname{Index}
\examplename{Example}
\keywordsname{Keywords}
\referencesname{References}
\endinput
```

The restrictions

are that as part of the replacement text tags are excluded which are not allowed in an argument of a `\def`.

2.3 Operations

What operations do we need? With databases we have the structure definitions of the fields, to fill these in via a fill-in screen, the various queries—read operations—next to the reporting features. I defined already the structures of the elements. My operations are

- adding to the database goes without fill-in screens but just by extending the file[6]
- as queries I provided: selective loading,[7] and making a list of names
- the reporting is the printing of the database, in general, a part of the typesetting task.

Adding to the database

comes down to extending the file with the list element.[8]

Selective loading

The loading is namelist driven, that is, the names of the entries to be loaded have to be specified.

For class I databases the markup for the loading reads

$$\backslash\langle kindofdatabase\rangle\{\backslash\langle name_1\rangle\ldots\backslash\langle name_n\rangle\}$$

with `<kindofdatabase>` either addresses, pictures, or references.[9]

For class II databases the markup for the loading reads for example

`letter` or
`report` or
`transparencies`

for formats. For tools it is tool dependent, for example for typesetting Pascal it comes down to

```
\beginpascal
<pascaltext>
\endpascal
```

with the details of loading hidden from the user.

Listing of all the names

can be obtained after, for example

```
\contentsdatabase{address}
%or
\contentstoolsorfmt{fmt}
```

Listing of names selected via pattern matching

In order to browse the class I databases, especially address.dat and lit.dat, the macro `\search` has been provided in blue.tex. The idea is that as input we specify a pattern—without periods, alas—and as result a list of names will be obtained, of which the replacement text contains the pattern. Those names and associated replacement texts are stored as definitions, i.e., the name will become the control sequence of the definition.[10]

Example *(Search for addresses from RUSSIA)*

Below the input has been given for the search of the pattern RUSSIA in the database file address.dat.

```
\input blue.tex
\searchfile{address}
\search{RUSSIA}
\bye
```

This is handy for making address labels grouped per country.
`\input blue.tex \letter`

[6] Perhaps someone can provide the screen fill-in facility on top some day?

[7] To load all the items is trivial, but generally too much. The gains in storage which can be obtained by selective loading have been reported in 'BLUe's References, revisited.' It is convenient to specify the elements by name.

[8] A white lie, more has to be done.

[9] Specifying by pattern will be treated a little later.

[10] This is precisely the reason to supply curly braces in the database entry.

```
\searchfile{address}
\search{RUSSIA}
\makesearchlabels
\bye
```

A search pattern should not contain a period.

Listing of contents

There are various kinds of listings, such as

- verbatim (format and tools, total or selective), and
- typeset (address labels, total bibliography).

Verbatim listing of a tool.

This problem is induced by my multi-file approach. Of course one can extract the tool by an editor et cetera as alternative.

Example *(Verbatim listing of macros of pascaltool)*

In order to do these kinds of things I have added tags as comments to the file. For example `%!cgl;newcol`, with the intention to use `;newcol` for a new column in a pfile listing and as an end separator with `\cgl` to terminate reading. Quite confusing but once understood it is handy to have those hooks.

```
\input blue.tex \let\pascaltool=x
\long\def\tool#1{\ifx#1\undefined
   \bgroup\unouterdefs\ea\gobbletool
   \else\ea\toolverbatim\fi}
\def\cgl;newcol{\endverbatim\endinput}
\input tools.dat
\bye
```

Remark. For a convenient verbatim listing of fmt.dat or tools.dat use pgfile.tex, which is a generalization of pfile.tex to print verbatim blue.tex.[11]

```
%pgfile.tex cgl, March 1995
\input blue.tex \hfuzz=25pt
\lineskip1pt plus2pt minus1pt
\baselineskip12pt plus 2pt minus1pt
\message{Type name for file:}
\read16 to\namefile
\title{File: \namefile}
\issue{Version 1.0}
\beginscript
  \thisverbatim={\catcode`\;=0
  \catcode`\!=12
  \catcode`\|=12
  \input \namefile}
  \beginverbatim
  ;endverbatim
\endscript
```

A convenient 'listing' of the address database is a list of address labels.

Example *(All address labels)*

```
\input blue.tex \letter
\makealllabels
\bye
```

3 Use

The use has been treated by example and is part of the larger typesetting task. For class I databases the use after specification reads as follows.

```
\pasteupreferences                    references
\<picturename>                         a picture
\letterto{...}          address(es) & letter.tex
```

More details about the use of `\letter` and pictures are provided in the user guide 'Publishing with TEX,' or more specifically in 'BLUe's Letters' and 'BLUe's Graphs.'

Example *(Typesetting letter(s))*

I assume that the letter proper has been stored in letter.tex.[12]

```
\input blue.tex \letter
\subject{\TeX{} for BLU}
\ourreference{22 1 95}
\yourreference{\TB}
\letterto{\knuthde\ntg}
```

Example *(Typesetting references)*

```
...
\references{\knuthdea\laancgc}
\beginscript
...
\pasteupreferences
\endscript
```

4 Miscellaneous

4.1 Testing for integrity after extension

Because of the restrictions it is advisable to test a database for use when it has been extended. The simplest test I could think of is to provide the 'table of contents,' that is a list of all the names. The test job for lit.dat reads as follows.

```
\input blue.tex
\contentsdatabase{lit}
\bye
```

No pages of output will be produced (that is OK!). The file contentslit will contain the list of all names. When this job is successful we know that TEX will not complain while scanning.

Note that in blue.tex I provided `\newwrite \toclit` and the like.

4.2 Extension with a format or tool

With tools and formats we have to work a little harder. In order to extend the database of class II, the following must be done.

- provide for user-interface macros as part of the kernel
- guard against multiple loading

[11] Within the context of verbatim printing I use the semi-colon as escape character.

[12] It is also possible to supply the letter in the job, but that is not so relevant for the database aspects.

- allocate storage in the kernel, blue.tex, i.e., only once.[13]

Note that outer definitions can't be used as such.[14]

Example *(Addition of the report format to format.dat)*

In order to achieve the use of `\report` we have to add to blue.tex the following overhead, and to include in fmt.dat the macros proper, preceded by `\tool\reportfmt` and ended by `\endinput`.

```
%User interface in blue.tex
\def\report{\ifx\undefined\reportfmt
   \let\reportfmt=x\fi
   \ifnum\reportloadcnt=0 \ea\loadformat\fi
   \advance\reportloadcnt1
   \let\reportfmt\undefined}
%with auxiliaries
\def\loadformat{\input fmt.dat\relax}
%Storage allocation
\newtoks\chapternumbering
%et cetera
%And in fmt.dat the blue collar workers
\tool\reportfmt
<reportmacros>
\endinput
```

I did not supply a message when `\report` was already defined, on purpose.[15]

Note that I did not talk about the design of `\report`. The redesign and coding of `\beginscript`, for example, is another issue, and beyond the scope of this paper.

Conventions for entries of fmt.dat

come down to

- update the contents at the beginning of the file fmt.dat
- include in the beginning of the format `\message{ ---<...> format, <date>, <owner>--- }`
- adhere to visual layout of the code for the printout
- insert additional comments which contain markup for the line numbering
- provide of table of contents with line numbers
- add a history of changes log.

The layout is aimed at printing in two columns, with a new contribution starting a new column. The selective line numbering is controlled by the control sequences

- `\numvrb`, to start the line numbering
- `\vrblin` = `<number>`, to adjust the line counter, and
- `\nonum`, to switch off the line numbering.

Do remember that within the context of a printout the escape character is the semi-colon.

Example *(Template for an entry in fmt.dat)*

```
%end%<lastlinepreviousformat>     %;newcol
%
%
%begin%%%%%%%%%%%%%%%%%%%%%%%%%<...>%;numvrb
\tool\<...>fmt
\message{ ---<...> format, <date>,
                      <(cr)owner>--- }
%<authorinformation>
<theformatproper>
\endinput                         %;nonum
%Contents
%<item>...................<pagenumber(s)>
%etc
%History of changes
%<date> <change>
%etc
%end%%%%%%%%%%%%%%%%%%%%%%%%%<...>%;newcol
```

A similar template holds for tools.dat. See 'Publishing with TEX,' for the details.

5 Coding

In BLUe's format two-part macros are the starting point. A one-part macro is provided on top of it. That is one of the conventions I adopted, relevant for the coding of the handling of databases.

The two classes of databases have a different user interface, and therefore the coding proper is class-specific.

5.1 Addresses, references, and pictures

For specification of the addresses, the references or the pictures, assign the names to a toks variable.[16] The loading from the database is for all three applications the same. However, what we do with it further depends on the application. Just to give you a break I'll digress a little on the latter.

Pictures are special, because we also need to load the common macros for use within any picture, assumed we use pictures at all.

Addresses have as extra that they have to be merged with one or more letters, and with the background material such as logo, sender affiliation and the like.

With references the extra work comes from the need for cross-referencing, and to paste them up at a place at will.

In this note I'll restrict myself to the loading of list elements and leave open what to do with these elements in the script. The latter is treated in the specific articles 'BLUe's Graphs,' 'BLUe's Letters,' and 'BLUe's References.'

The coding for loading elements from a database contains the two major steps[17]

[13] Remember that TEX lacks a garbage collector.

[14] Replace the definition by a non-outer one, or use `\csname`. One can also add the def or symbol to the set of `\unouterdefs`.

[15] I usually start separate chapters with `\report` and then I don't like to have to remove the `\report` tags in the chapters, when the total will be processed. Nor do I like to get messages, because I know already that the tag is at the start of each chapter. Of course reloading is prevented.

[16] Well, . . . a white lie, it looks like it.

[17] Because of the freedom to give it any meaning it is handy to let the names stand for the error message at first. In case of a typo in the name it already contains its error message. Neat!

- define the specified names
- load the definitions associated with those names.

In other words, in contrast with the usual procedure — if something is already known don't load it again—the opposite has been applied here. When a name is known it will be overloaded, i.e., replaced by what we really want. As extra I created a `\namelst` with the specified names each preceded by `\lst`.[18] As abstraction I used in the coding below the root name 'kind of database, KoD for short' which could equally well have been addresses, pictures, or references, in principle.

```
\def\<KoD>#1{\begin<KoD>#1\end<KoD>}

\def\begin<KoD>#1\end<KoD>{%
  \let\process\processnames\fifo#1\ofif
  \loadselectivefrom{\the\<KoD>file}}

\def\processnames#1{\ifx\undefined#1
    \let#1\<KoD>error
  \else\message{\Dash\string#1
         already loaded.\Dash}%
  \fi\namelst\ea{\the\namelst\lst#1}}

\def\loadselectivefrom#1{%#1 <name>
 \def\lst##1{\ifx##1\undefined\ea\gobble
             \else \ea\gdef\ea##1\fi}
 \input #1.dat \relax}%e.g. lit.dat

\def\<KoD>error{listelement not
    in database, (Sorry.)
    \loaderror{<KoD>}}

\def\loaderror#1{\message{#1 not in
      database}}
```

Explanation. Remember that the file consists of triples: `\lst`, `\<name>`, and a group, 'the replacement text.' Are you ready? There we go. All the specified names are defined and obtain first their error message as replacement text. This is done via the use of the fifo paradigm. The loading overrides each (error message) replacement text with what we want. This is done via looking at `\lst` as the list element tag with as argument `\<name>` with the purpose to store or to gobble the third element of the triple.

As extra the names are stored in `\namelst`, each preceded by `\lst` to facilitate later processing, i.e., without recursion, just execute the `\namelst`, with an appropriately defined `\lst`.

The one-part macro for this simple case is trivial and added for consistency. The argument does not need processing on the fly.[19]

5.2 Formats and tools

For the loading it comes all down to what we assign as meaning to the list element tag, that is, what meaning we assign to `\tool`. In the same spirit as with references and the like, it comes down to gobble the third element of the quadruple if the `\<name>tool`, respectively `\<name>fmt`, is unknown. Otherwise all the stuff up to `\endinput` will be loaded, and that is it.

The entries from format.dat are not mentioned below, because that is not relevant any longer. They just have to obey the earlier given syntax.

General macros and basic tools for selective loading of formats are the following, borrowed form blue.tex.

```
\def\gobbletool#1\endinput{\egroup}
%
%Define the list element tag '\tool'
\long\def\tool#1{\ifx#1\undefined
   \bgroup\unouterdefs\ea\gobbletool\fi}
%
\def\unouterdefs{%List of defs which
                 %have to be neglected
  \ea\let\csname+\endcsname\relax
  \catcode`\^12
}
%
\def\toolverbatim{\thisverbatim{%
    \baselineskip12pt plus2pt minus1pt
    \lineskip1pt plus1pt minus1pt}%
    \beginverbatim}
%
\def\loadformat{\input fmt.dat\relax}
```

Note that `\unouterdefs` have been kept local.

The user-interface macro for `\letter`, as provided in blue.tex, reads as follows.

```
\def\letter{\ifx\undefined\letterfmt
   \let\letterfmt=x\fi
   \loadformat
   \let\letterfmt\undefined}
```

Multiple loading has been safe-guarded in `\report`, via the use of the counter `\reportloadcnt`, as follows.

```
\newcount\reportloadcnt
%Prevent double loading.
\def\report{\ifx\undefined\reportfmt
   \let\reportfmt=x\fi
   \ifnum\reportloadcnt=0 \ea\loadformat\fi
   \advance\reportloadcnt1
   \let\reportfmt\undefined}
```

5.3 List of names in the database

The following macro does the job for the class I databases. It creates a file contentsaddress, contentslit, or contentspic, with the list of all the names.

```
\def\contentsdatabase#1{%#1 address lit pic
\ea\let\ea\name\csname toc#1\endcsname
\immediate\openout\name=contents#1
\def\lst##1##2{\immediate\write\name{\nx##1}}
\input #1.dat\relax}
%with auxiliaries
\newwrite\tocpic
\newwrite\toclit
\newwrite\tocaddress
%test example
\contentsdatabase{address}
\bye
```

[18] I did not want to bother the user with preceding each name by `\lst`. I chose for alleviating the task for the user.

[19] Because I chose to typeset `\listelementerror` when not overwritten, it must obey the syntax of the typesetting macro.

Note that in order to write the (undefined) names no expansion must take place.

For databases of class II—with list element tag `\tool` and end separator `\endinput`—the following does the job. It creates a file `contentsfmt`, or `contentstools`, with the list of all the names.

```
\def\contentstoolsorfmt#1{{%#1 tools fmt
\ea\let\ea\name\csname toc#1\endcsname
\immediate\openout\name=contents#1
%Define list element tag '\tool'
\long\def\tool##1##2\endinput{%
    \immediate\write\name{\nx##1}}
\unouterdefs
\input #1.dat\relax}}
%with auxiliaries
\newwrite\tocfmt
\newwrite\toctools
%test example
\contentstoolsorfmt{fmt}
\bye
```

Note that in order to write the (undefined) names to a file no expansion must take place.

5.4 Search by pattern

Searching a database for a pattern with the aim to yield a list of names—each name preceded by `\lst` in the toks variable `\namelst`—has been coded as follows.

```
%Searching the database
\def\search#1{\def\loc##1##2{%
   \def\locate####1##1####2\end
    {\ifx\empty####2\empty\foundfalse
     \else\foundtrue\fi}%end locate
   \ea\locate##2.##1\end}%end loc
   \def\lst##1##2{\loc{#1}{##2}\iffound
    \immediate\write16{\nx##1}%log file
    \namelst\ea{\the\namelst\lst##1}\fi}
   \input \the\searchfile.dat\relax}
%with auxiliaries
\newtoks\searchfile
\newtoks\namelst
```

Explanation. The argument of `\search` is matched with the replacement text of each list element. When found the name of the list element is added to the token variable `\namelst`. For the time being the name is also written to the log file, for the purpose to verify what has been added to the token variable.[20]

For a pedagogical stepping stone see 'Syntactic Sugar' where I used the `\loc` macro to locate a letter in a string, and applied it to Appelt's problem of doing something special to capital characters.

6 Looking back

It all started with my surprise when studying AMS styles and formats that references have to be marked up over and over. The use of databases was badly needed. So I started to think about why it has been overlooked. I realized that from a database only a few references will be included in an article or so. On the other hand, the idea struck me that an author should not bother about formatting references at all. He should supply only names to entries already available, and appropriately marked up, in the publisher's database.

The overhead of other tools in use by the community was clear to me, so I started to work on 'BLUe's References.' During that work I found how to load selectively from a TEX database. That was a gem, a paradigm, and bound to be used elsewhere. Since then, especially while working on 'BLUe's Format,' I have used the mechanism for addresses—to be merged with letters—for pictures—to be stored scaling invariant, and to be invoked with a particular `\unitlength`.

When I faced the problem of how to store elegantly formats and tools in general, I combined my[21] selective loading approach with Knuth's lists. Et voilà.

7 Looking forward

I don't know as yet to what extent the above is useful for databases which are an order of magnitude larger, i.e., with thousands or tenthousands of entries.

8 Acknowledgements

Thank you Jos Winnink for your interest in the database approach with TEX, and for stimulating me a bit for writing this separate article on the issue.

9 Conclusion

It is so nice to watch the TEX log file scroll by slowly, witnessing the searched databases, smoothly and on-the-fly.

References

The TEXbook and LATEX user's guide are omni-present and not explicitly listed. For my works consult lit.dat, via the use of `\search` for example.

[20] The searching for a pattern within TEX has been used by me on several occasions already. To start with typesetting bridge, where the purpose was to locate a card in a hand, and if so to delete the card, that is, update the hand. This was a special case of determining whether an element belongs to a set. The latter occurred further in determining whether a token belongs to the set of accents, whether a token belongs to the set of reserved words of Pascal, and the like. Sooner or later it had to pop up in pattern matching.

[21] Well, . . . it has been essentially in *The TEXbook* 384 all the time. I'm only wondering how much there is still left unnoticed.

BLUe's Index

The principle and some more

Kees van der Laan

Hunzeweg 57,
9893 PB Garnwerd, The Netherlands
cgl@rc.service.rug.nl

Abstract

The creation of a modest index within a one-pass TEX job is proposed. In general a proof run and a final run are needed. The markup for the index entries is the same as used by Knuth for *The TEXbook*. The process is controlled by the tags: `\sortindex` and `\pasteupindex`. The file which emerges after `\sortindex` can be enriched by hand, for example to include 'see also' and the like. Sorting keys can be specified too, as well as items which have to be ignored for the sorting. The macros have been developed for use with English documents. To leave open the use with other languages the ordering has been parameterized in an ordering table.

Keywords: Compatible extension, education, index, macro writing, number ranges, one-pass job, ordering table, plain TEX, reusable software parts, software engineering, sort keys.

1 Introduction

Making an index is an art. The fundamental problem is

What to include in the index?

Computer-assisted indexing is not simple either. Issues are

- the markup of keywords or phrases
- to associate page numbers
- to sort and compress raw Index Reminders (IRs), and
- to typeset the result.

The latter opens the Pandora box of markup and layout, which prompts for a two-pass job.

Up till now the sorting is done outside of TEX.[1] However, producing a modest index in a one-pass TEX job is possible. In 'Sorting in BLUe,' I already touched upon the issue of sorting IRs.

I'll build upon manmac's[2]

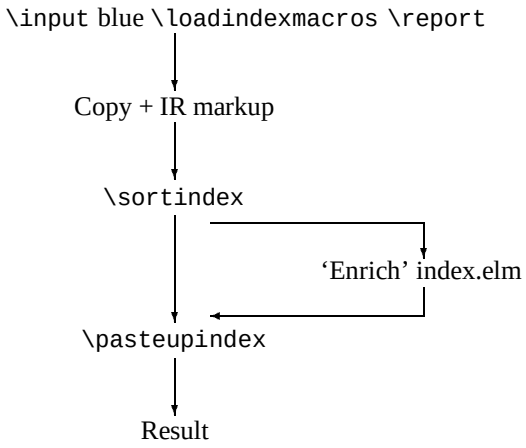
- syntax and types of Index Reminders (IRs)
- writing to a file
- associating page numbers, and
- the formatting in two-columns.

2 The process and files involved

Manmac stores the raw IRs in the file index. I read the file index[3] into an array for internal sorting. After sorting I reduce the entries[4] and write the result to the file index.srt. Then, I transform index.srt into the file index.elm.[5] The result is typeset via `\pasteupindex`.

2.1 Schematically

It comes down to



[1] The index for the TEXbook was done by a multi-pass job. In proofmode the raw IRs are written to the file index, and in the final run the enriched file of index entries is encluded in the script, preceded by special markup definitions.

[2] In contrast with Knuth I refrained from the full stop of each entry in the typeset index.

[3] Default index is the value of the toks variable `\irfile`, which is used in `\sortindex`.

[4] Those which differ by page number are collected into one entry.

[5] Default index.elm is the value of the toks variable `\indexfile`, which is used in `\pasteupindex`. The transformation abandons the IR syntax. The part which specifies the kind of IR is deleted and the word part marked up accordingly.

3 Why?

For a professional index I agree with Knuth that one can better sort and otherwise enrich the index outside of TEX. This obeys the separation of concerns principle. However, there is nothing against it to provide macros for producing modest indexes completely within TEX. Objections[6] will faint away hopefully, because of the possibility to *enrich the sorted and compressed index (file) outside of TEX*. The latter is a step beyond manmac. There the raw IRs are written to the file index, while here the sorted and compressed IRs can be worked upon. My work is a compatible extension of manmac, meaning that one can also start from the raw IRs, if one wishes.

A side-effect is that the power of TEX for practical problems is demonstrated.

4 Disclaimer

The macros are not foolproof, :-). Don't use as part of the IR

- TEX's special symbols
- symbols with special catcodes, for example active symbols like the circumflex ^, or the tilde ~
- as part of the IR a control sequence which is not accounted for.[7]

5 Notations and definitions

`manmac.tex` stands for the macros used by Knuth for formatting his Computers and Typesetting series of books, especially the TEXbook and the METAFONT book. Index reminders denote the markup of script words, to be included in the index. IRs is an abbreviation for Index Reminders. `\ea`, `\nx` denote `\expandafter` and `\noexpand`. OTR stands for output routine. FIFO stands for First In First Out, as described in 'FIFO and LIFO sing the BLUes.' SiB denotes my 'Sorting in BLUe' work.

6 Example of use: From the TEXbook

Let us take for this example some IRs from the first chapter of the TEXbook. I took only part of it and introduced `\newpage` now and then to enforce pages with zero, one or more IRs. I used `blue.tex` (with manmac embedded).

```
\input blue.tex \loadindexmacros
%Text from first chapter TeX book
Hence the name \TeX, which is an
uppercase form of $\tau\epsilon\chi$.
^^{TeX, meaning of}%modified
^^|\tau|^^|\epsilon|^^|\chi|
\newpage%no IRs next
Insiders pronounce the $\chi$ of \TeX\ as
a  Greek chi, not as an 'x', so that
\TeX\ rhymes with the word blecchhh.
\newpage%three IRs next
In fact, ^{TEX} (pronounced {\sl tecks\/})
is the admirable {\sl Text EXecutive\/}
processor developed by^{Honeywell HIS}.
Since these two system names are
^^{Bemer, Robert}
pronounced quite differently, they should
also be spelled differently.
\newpage%one IR next
The correct way to refer to \TeX\ in a
computer file, or when using some other medium
that doesn't allow lowering of the 'E',
is to type '^|TeX|'. Then there will be no
confusion with similar names, and people will
be primed to pronounce everything properly.
\sortindex\pasteupindex
\bye
```

which yields as raw IRs (in file index)

```
TeX, meaning of !0 1.
tau !2 1.
epsilon !2 1.
chi !2 1.
TEX !0 3.
Honeywell HIS !0 3.
Bemer, Robert !0 3.
TeX !1 4.
```

and as index (in file index.elm)

7 Example of use: Accents and the like

I'll show how to mark up Knuth's four types of IRs and how to mark up accents, font switching, and spaces as part of the IR. The last IR markup shows that the representation of page numbers as a range comes out automatically too.

```
\input blue.tex \loadindexmacros
Types of IR
 0  ^{return}
 1  ^|verbatim|
 2  ^|\controlsequence|
 3  ^\<syntactic quantity>

Accents in IR markup  ^{\'el\`eve!},
font changing in IR markup ^{\bf bold}
and spaces in IR markup ^{control\ symbol}

Control sequences ^{\TeX, and \AmSTeX}
^{Lamport and \LaTeX}

brackets ^{\tt< \rm and \tt>}
\newpage range representation ^{return}
\newpage                      ^{return}
\sortindex\pasteupindex
\bye
```

The above yields in file index as raw IRs

```
return !0 1.
verbatim !1 1.
controlsequence !2 1.
syntactic quantity !3 1.
```

[6] If any :-).

[7] In the section Looking forward I'll explain how a user can allow for control sequences as part of the IR markup.

```
\'el\'eve! !0 1.
\bf{}bold !0 1.
control\ symbol !0 1.
\TeX, and \AmSTeX{} !0 1.
Lamport and \LaTeX{} !0 1.
\tt{}< \rm{}and \tt{}> !0 1.
return !0 2.
return !0 3.
```

and in file index.elm as index

8 Index Reminders

IRs are at the heart of the process. Knuth distinguished 4 types to facilitate the outside processing. I'll adopt his IRs syntax and types.

8.1 Syntax

Knuth's IRs obey the syntax[8]

⟨*word(s)*⟩ `!`⟨*digit*⟩ ⟨*page number*⟩`.`

The digits 0, 1, 2, or 3 denote the types: words, verbatim words, control sequence, and syntactic quantity. A user does not have to bother about the digits, nor about the page numbers. Knuth has adopted the accompanying conventions for the word(s) of IRs.[9]

Mark up	Typeset in copy*	IR
`^{...}`	...	`...` `!0` ⟨*page no*⟩`.`
`^|...|`	`|...|`	`...` `!1` ⟨*page no*⟩`.`
`^|\...|`	`|\...|`	`...` `!2` ⟨*page no*⟩`.`
`^\<...>`	⟨...⟩**	`...` `!3` ⟨*page no*⟩`.`

* `|...|` denotes manmac's, TUGboat's, ... verbatim.
** in `\rm`.

For the user the word(s) are important. The allowed markup for the IRs and the result in the copy are given in the accompanying table.

8.2 Markup

The markup for IRs is near to natural. Precede the entry by a circumflex (or two circumflex(es) in case of a silent index entry).[10]

Example *(IR markup.)*

```
^{text, e.g. \'el\'eve!}
^|verbatim text|
^|\controlsequence|
^\<a metalinguistic variable>
%and for silent ones, double the ^
^^\<a metalinguistic variable>
%from the TeX book
{\sl^{ligatures}}
|'$|^|\,||$''|
^^{markup commands, see control sequences}
%from Looking forward section
^{Lamport and \LaTeX}
```

Spaces

are as always difficult. In the IR they separate parts of the IR, and are used in the word part.

Just typing a space has as effect that it will be neglected during sorting.

The markup '`\␣`', a control space, will yield a space subject to sorting, according to the ordering table.

`\space` markup will be neglected during sorting. This token is default member of the set of to be ignored control sequences. It will be set in the index as `\␣`.

Example *(Spaces)*

The following markup

```
\input blue.tex \loadindexmacros
Spaces test
^{\space}%an ignored cs
^{a\ a}  %control space
^{aa}
^{a\ b}
^{a \TeX}
^{a\ \bf a}
^{\TeX book}
^{xyz beta}%space neglected in sorting
^{xyza}
^|\space|
\sortindex\pasteupindex
\bye
```

yields as file index

```
\space {} !0 1.
a\ a{} !0 1.
aa{} !0 1.
a\ b{} !0 1.
a \TeX {} !0 1.
a\ \bf a{} !0 1.
\TeX book{} !0 1.
xyz beta{} !0 1.
xyza{} !0 1.
space{} !2 1.
```

and as file `index.srt`

```
\space {} !0 1.
a\ \bf a{} !0 1.
```

[8] In contrast with my previous given syntax it seems that Knuth was less restrictive. Earlier I overlooked that the entries are ended by a period.

[9] See *The TeXbook* 424 for the IR types, and what is typeset in the result. In `\vref` the markup is inserted as replacement text of `\next`. What is set in the index is governed by the macros which are included after `\begindoublecolumns` in the *The TeXbook* script.

[10] Silent IRs mean that these will appear only in the index, not on the page.

```
a\ a{} !0 1.
a\ b{} !0 1.
aa{} !0 1.
a \TeX {} !0 1.
space{} !2 1.
\TeX book{} !0 1.
xyza{} !0 1.
xyz beta{} !0 1.
```

Explanation. \space belongs to the set of to be ignored control sequences, ICSs for short. This means that it is skipped with respect to sorting, except when it occurs as the last token of the word part. In that case they are ordered as a space, that is according to the lowest value. This explains the position of '\space.'

'\TeX,' and '\TeX book,' are subject to the default sort keys.

'xyza' precedes 'xyz beta,' because the space is silent. When word ordering is preferred a \␣, a control space, must be included.

8.3 Writing the IRs in index

comes with manmac. The writing is done in two phases: first while processing the script, and second in the OTR where the page numbers are attached. The (manmac) macro which does the first part of the writing is

```
\def\makexref{\ifproofmode
   \bgroup\def\ {\string\ }%
   \xdef\writeit{\write\inx{\text{}
    !\xreftype\space
    \nx\number\pageno.}}\writeit
   \egroup
   \else\ifhmode\kern0pt\fi\fi
   \ifsilent\ignorespaces\else\next\fi}
```

I don't write in the margin. In Appendix C the full-BLUe macro has been provided. Font changing control sequences, accents and \space are written as a 'string' in the file index. Actually, I introduced also the sorting on sorting keys. Because of the various uses of \space I introduced \spaceseparator.

9 Sort and compress: \sortindex

The functionality of \sortindex is to transform the file of raw index reminders — index — into the file with sorted and compressed index entries — index.elm.

All we have to do is to

- write the raw IRs in the array, and keep the upper bound of the array in \n,
- sort with the right comparison macro (\cmpir, and at the lower level for the word part \nxtwindex), next to the use of the ordering table \otindex
- reduce the index entries by collecting the same entries which differ by page number
- write the result to the file index.srt, and transform this into index.elm.

```
\def\sortindex{%Nov 1994, cgl
%Purpose:
%To sort IR file.
%Input: default index is sorted
%      (file specified in \irfile)
%Output: file index.elm.
\newpage\immediate\closeout\inx
\filetoarray{\the\irfile}
\immediate\write16{Sorting n=\the\n.
    Please wait, O(nlog n) process.}
\let\cmp\cmpir\let\nxtw\nxtwindex
\otindex\let\ \space
\sort
{\let\spaceseparator\space
 \setupnxtokens\def\ {\nx\ }%
 \immediate\write16{Range reduction.}
 \redrngtofile{index.srt}
 \immediate\write16{After reduction and
  writing to file index.srt; n=\the\n.}
 \immediate\write16{Transform
       index.srt-->index.elm.}
 \tawfiletofile{index.srt}{\the
                  \indexfile}}}
```

with auxiliary \def\setupnxtokens{%

```
 \def\process##1{\def##1{\nx##1}}%
  \ea\fifo\the\conseqs\ofif
 \def\process##1{\def##1{\string##1}}%
  \ea\fifo\the\consyms\ofif
}
```

The sorting within TEX has been treated in SiB, and will not be repeated in this article. A user must supply a comparison macro.

In Appendix C the full-BLUe version has been incorporated.

9.1 Storing in an array

The storing of data from a file into an array has been treated in SiB. A slightly modified version of the macro reads

```
\def\filetoarray#1{%#1 is file name
 \immediate\openin\inxin=#1\relax
 \ifeof\inxin\immediate\write16{File #1 empty
   or non-existent.}%
 \fi \n\kzero\continuetrue
 \loop\ifeof\inxin\continuefalse\fi
 \ifcontinue\advance\n1 \immediate
 \read\inxin t\ea o\csname\the\n\endcsname
 \repeat\advance\n-1
 \immediate\closein\inxin}
```

9.2 Comparison

For sorting I make use of my macros as released in SiB. Comparison needs a multiple key. This means that at the outer level we have to supply \cmpir[11] and at the lower level we have to compare words, and digits. The words are handled by \nxtwindex and the numbers are compared via an \ifnum. The comparison macro for IRs reads

```
\def\cmpir#1#2{%#1, #2 defs
%Result: \status= 0, 1, 2 if
%        \val{#1} =, >, < \val{#2}
\ea\ea\ea\decom\ea#1#2}
```

The crucial macro for comparison of the word part of the IR reads

```
\def\nxtwindex#1#2{%
```

[11] Mnemonics compare index reminder.

```
%Function:
%On input: #1 contains the 'word'
%As result: #2 contains the value
%   of the first non-ignored token as given
%   in the ordering table.
%#1 contains the rest of the 'word'
 \def\pop##1##2\pop{%
 \gdef#1{##2}\def\pophead{##1}}%head and tail
 \ea\pop#1\pop%split in head and tail
 \ignores\ea\ea\ea{\ea\the\ea\conseqs
                        \the\consyms}%
 \ea\loc\pophead{\the\ignores}%
 \iffound\ifx\empty#1 \chardef#2=0
        \else\nxtwindex#1#2\fi
 \else
 \ea\let\ea#2\csname ot\pophead\endcsname\fi
}
%with toks variables
\conseq{\c\space\bf\it\rm\tt\TeX\sub}
\consyms{\'\'\"\^\~}
```

The idea is to process the 'word' argument by argument.[12] However, some tokens are not relevant for the ordering and have to be ignored. This is done by collecting all tokens to be ignored in the toks variable `\ignores`, and compare `\pophead` with this string. The above can be extended to control sequences to be sorted on separately provided sorting keys. See the Looking forward section.

For `\decom` and other low level macros related to sorting within TEX see 'BLUe's Format,' or 'Sorting in BLUe.'

In Appendix C the full-BLUe versions have been incorporated.

Ordering

A fundamental issue with indexes is the ordering. The ASCII table is not suited because lowercase and uppercase letters differ by 32. I decided to rank these as equal, more precisely to assign the lowercase ASCII values to both. I prefer from the following the left column above the right one

```
el                      el
Elève                   em
em                      Elève
```

Moreover, accented letters are not part of ASCII. How should we order for example e, é, è, ê, ë? I decided to rank accented letters equal to those without an accent, because I prefer from the following the left column above the right one

```
el                      el
élève                   em
em                      élève
```

I know that non-letters precede letters but what about there relative ordering? I decided to stay as close as possible to the ASCII ordering.

Then there is the problem of digits. In IRs they come as part of the word(s) and as page numbers. For the latter I used the numerical ordering. For the former I used the alphabetical ordering.[13]

Furthermore, a user can select the so-called 'word ordering,'[14] by `\␣`, TEXnically a control space, as markup for a space. Personally, I like from the following the first column better than the second

```
sea lion                seal
seal                    sea lion
```

Ordering table

This ordering 'table' is simpler than the one released in SiB, because accents have been ignored. Furthermore, I adhere mostly to the ASCII ordering, as can be seen easily.

```
\def\otindex{%Parameters: Ordering 'table'
%Special cases
\ea\chardef\csname ot \endcsname=0
\ea\chardef\csname ot\space\endcsname=0
%{|}~ come in ASCII after lowercase
%^ is active character
%Bulk according to ASCII
\def\process##1{\ea\chardef
   \csname ot##1\endcsname='##1 }
%lowercase letters
\fifo abcdefghijklmnopqrstuvwxyz\ofif
\chardef\otij='y \chardef\otIJ='y
%other characters}
\fifo!"##$&'()*+,-./0123456789:;<=>?@
      []_`\ofif
%assign lowercase values to uppercase letters
\def\process##1{\ea\chardef
  \csname ot##1\endcsname=\lccode'##1 }
\uppercase{\fifo
  abcdefghijklmnopqrstuvwxyz\ofif}
}
```

Attention needs TEX's specials, in particular the escape character \, the circumflex ^, and the percent %.

9.3 To be ignored tokens

In practice I needed things like `\tt` as part of the IR, which must be neglected while sorting.[15] I decided to ignore those tokens while sorting and to include the tokens in the final index.elm as such.

Another realistic approach is to add this kind of markup later in the file index.elm.

9.4 Reduction of entries

The functionality is that the IRs which differ by page number are collected into one entry with the page numbers represented efficiently, for example in ranges. The macros from SiB have the functionality

```
\def\redrngtofile#1{%Reduction of \1...\n,
%with range representation of page numbers.
%The result is written to file #1.
...
}
```

[12] Not token by token, beware!

[13] I could have applied a look ahead mechanism and use numerical ordering throughout. Maybe another time.

[14] This means that spaces precedes all letters. A space as such is neglected in the ordering.

[15] The reason is that `<, and >` are used and then printed wrongly.

```
%
\def\prcrng#1{%Prints the numbers so far if
%the new number differs more than 1 from the
%last. If the difference is 1 the range is
%extended.
...}
%
\def\strnrs{%Accumulates numbers in \nrsrng.
%either as a range or as such.
%\frst stands for first number
%and \last for last number. If they equal the
%number is stored, if they differ by 1 both
%the numbers are stored separated by \sepn,
%and if they differ by more than 1
%\frst--\last is stored.
...}
```

For the macros see Appendix C.

9.5 Transformation index.srt→index.elm

When we inspect the copy for the index in the TEXbook file then we'll find that the entries of the enriched file don't obey the syntax of the IRs. The part with !⟨*digit*⟩ has disappeared. Pondering about this made me agree with Knuth, as usual.[16] It is no longer functional! The file can better be considered as copy as such, to be processed in the final run. Because of this I transformed index.srt, the file of sorted and compressed IRs, into the file index.elm, with the coding !⟨*digit*⟩ absorbed as markup in the word part.

```
\def\tawfiletofile#1#2{% #1 from file
 \continuetrue         % #2 to file
 \immediate\openin\inxin=#1\relax
 \immediate\openout\inx=#2\relax
 \loop\read\inxin to\IR
   \ifeof\inxin\continuefalse\fi
 \ifcontinue\trfandwrite\IR
 \repeat
 \immediate\closein\inxin
 \immediate\closeout\inx
}
%with auxiliaries
\newread\inxin\newwrite\inx\newtoks\indword
\def\splitintoks#1 #2 #3.{\indword{#1}%
  \chardef\digit=#2\relax\def\pagenrs{#3}}
%
\def\trfandwrite#1{\ea\splitintoks#1%
 \immediate\write\inx{\nx\noindent
  \ifcase\digit{\the\indword}\or
   {\nx\tt\the\indword}\or
   {\nx\tt\char92\hbox{\the\indword}}\or
   $\nx\langle\hbox{\the\indword}\nx\rangle
    $\fi\spaceseparator{\nx
   {\oldstyle\pagenrs}}}
```

Explanation

In `\trfandwrite` I used the toks variable for storing the word part, because when writing it comes out handy that `\the` is a one-step expansion. Note that `\trfandwrite` takes care of the markup for `\pasteupindex`. The `\noindent` is inserted to enforce horizontal mode, especially in presence of accents at the beginning of the word. In `\tawfiletofile` the test for the end of file looks peculiar.[17]

10 Typeset: `\pasteupindex`

In the *The TEXbook* the typesetting of the enriched index entries is treated on 261–264. Let us start from there and distill our specifications.

The typesetting of main and subsidiary entries is discussed given the file of index entries. This is intertwinned with the page break mechanism in relation to what should appear in the running headlines.

My specifications for typesetting the index are

- represent the four IR types the same as in the TEXbook
- set in two-columns, balanced, possibly preceded by one-column copy
- set subsidiary entries analogous to the TEXbook
- indent continuation lines by 2em
- indent subsidiary entries by 1em
- underline page numbers which represent the definition or the main source of information
- represent a page number in italics when that page contains an instructive example of the concept in question.

Essentially nothing new. The challenge is how to implement this, such that an index can be handled in a one-pass job.

In order not to make things too complex, I'll postpone the handling of subsidiary entries until the Looking forward section. Let us say that this is a feature for the 'next release,' of `blue.tex`. The enrichements such as representation of numbers in italics or underlined, which have all to do with the wish to direct readers to the main source or to instructive examples, are left to the manual editing of the index.elm file. The reason is — In agreement with Knuth? — that this is difficult to foresee at the time when the IR markup is inserted in the script. Moreover, it complicates the sorting et cetera process.

In the mean time users can edit index.elm — read add markup — and provide the necessary macros in for example `\preindex`. In short follow Knuth.

The compressed and sorted aray of index entries is set via the invocation `\pasteupindex`.[18] The contents of `\preindex` and `\postindex` are used appropriately.

```
\def\pasteupindex{%Nov 1994, cgl
%Purpose:
%To set index in (balanced) doublecolumn.
%The index is preceded by contents of
%\preindex and followed by contents of
%\postindex.
%Input: default index.elm is set
%       (file specified in \indexfile).
```

[16] I noticed furthermore, that some IRs did not make it into the final index. Apparently, given the other IRs, he decided to delete less relevant ones.

[17] Remember that TEX appends a `\par` to the input file.

[18] As modification for the pair `\begindoublecolumns` and `\enddoublecolumns`, because the latter is too much intertwinned with manmac. For an explanation of the underlying macro the reader is referred to the *The TEXbook* 416–417.

```
%Biased by manmac's \begindoublecolumns
 \newpage\begingroup
 \def\space{{\tt\char32 }}%
 \the\preindex\par
 \pageheight\vsize
 \pagewidth\pagewd%anachronism
 \parindent1em
 \output={\global\setbox\partialpage=
  \vbox{\unvbox255\bigskip}}%
 \eject
 \output={\bluedoublecolumnout}%
 \hsize=8.5cm \vsize=51cm%blue.tex values
 \parskip0pt plus.8pt\relax
 \obeylines\everypar{%
    \hangindent2\parindent}%
 \let\par\endgraf
 \let\sub\endgraf
%
 \input\the\indexfile\relax
%
%endpasteupindex part biased by
%manmac's \enddoublecolumns
 \output={\balancecolumns}\eject
 \endgroup
 \pagegoal=\vsize\the\postindex
}
%
%auxiliaries adapted from manmac
\def\bluedoublecolumnout{%
%Biased by manmac's doublecolumnout
 \splittopskip=\topskip
 \splitmaxdepth=\maxdepth \dimen@=25cm
 \advance\dimen@ by-\ht\partialpage
 \setbox0=\vsplit255 to\dimen@
 \setbox2=\vsplit255 to\dimen@
 \blueonepageout\pagesofar
 \unvbox255 \penalty\outputpenalty}
%
\def\blueonepageout#1{%
%Biased by manmac's \onepageout
 \shipout\vbox{%
 \vbox to\baselineskip{\null
        \the\headline\vss}%
 \kern2ex
 \vbox to\pageheight{#1}%
 \kern1ex
 \the\footline
 }\advancepageno}
```

The file name is parameterized in a token variable `\indexfile`. Default is `\indexfile{index.elm}`. A user can specify his own file via

`\indexfile{`⟨*user index file*⟩`}`

10.1 Running headlines

The advanced mechanism of having the top entries and bottom entries appropriately represented in the headline has been treated in the TEXbook.

Looking at the indexes of the TEXbook and the more recent Graphbase book, I noticed that Knuth did not use it. In Appendix D he states that the index is not tall enough to justify the mark mechanism. It is really advanced and subtle, and for those who are interested, please peruse the TEXbook.

11 Customization

A user might want to interfere at the places

- to include other tokens to be ignored while sorting[19]
- to supply an ordering of his own
- to enrich the sorted and compressed file index.elm.

11.1 Adding tokens to be ignored

In general we don't know what control sequences stand for.

What are reasonable requirements to impose upon the handling of markup control sequences (cs for short)? In my opinion

- the cs must be defined
- `\makexref` writes the cs unexpanded
- ordering? unknown, and therefore neglected[20]
- `\setupnxtokens` guards that the cs-s are written, unexpanded, to index.srt and index.elm.

As a consequence I decided to neglect the 'in between' control sequences while sorting. For those who favour a one-pass job, I have provided the following, although it is simpler to add those control sequences to index.elm.

The extension of the set of to be ignored tokens can be done via

```
\def\add#1to#2{...}%See App C
%for example adding to set of control
%sequences
\add\hfil to\conseqs
%or control symbols
\add\'to\consyms
%Adding to the set of sort key pairs
\add\hfil{hfil}to\srtkeypairs
```

Each element from `\conseqs` is redefined such that the control sequence token is written to the file with a space appended.[21]

11.2 Modifying ordering

The most general way is to 'copy' the ordering table for modification.[22]

And what about a macro to add to the table?

This can be done easily, and superficially looks convenient for an innocent user. At the moment I don't trust the macros to be worthwhile for an innocent user, unless a very modest index has to be made. And this completes the circle: different ordering is not wanted, I guess.

11.3 Enriching the index

This use is necessary when for example

- control sequences have to be typeset
- special symbols are needed, or

[19] However, in the Looking forward section this has been generalized into the possibility to provide the control sequence with a sorting key.

[20] However, see the Looking forward section, ;-).

[21] `\noexpand` is used instead of `\string`.

[22] My `\fifo` is just a shortcut, which also prevents typos in assigning the ASCII values. For `\fifo`, see 'FIFO and LIFO sing the BLUes.'

- cross-references within the index are required.

The best way is to start from the index.elm file.

An example of use is that in the 4TEX manual the index is sorted on 4TeX, but in the index a different representation is desired. (The control sequence `\fourtex` has been used for typesetting instead of 4TeX.) For that purpose the file with IRs contains 4TeX and later this is substituted in the file index.elm by `\fourtex`.

Another approach is to supply pairs of control sequences and sorting keys. See the Looking forward section.

11.4 Typesetting the enriched file

When the default name is used — index.elm — just say `\pasteupindex`. For another file name assign this name to the toks variable `\indexfile`, prior to the invocation of `\pasteupindex`.

12 Tests

The macros have been tested on the file which was obtained from the TEXbook chapters 1 to 6. The (slightly adapted) file of raw IRs and the resulting sorted and compressed index entries have been included in the Appendices A and B. The test had to do with some 220 raw IRs.

A driver program, which starts from a file of IRs not necessarily generated by manmac, is

```
\input blue.tex
%\irfile{erik.15b}    %file with 6 entries
\irfile{erik.16b}     %file with 19 entries
%\irfile{indexdat.610}%file with 183 entries
%\irfile{erik.cgl}    %file with 228 entries
\sortindex\pasteupindex
\bye
```

Another test file was from the 4TEX manual. This makeindex file consisted of roughly 760 entries of the form `\indexentry{...}{<number>}`. I first transformed this file — by TEX — into a file obeying the IR syntax, via

```
\newwrite\erik
\immediate\openout\erik=erik.cgl
\def\indexentry#1#2{%
 \immediate\write\erik{#1 !0 #2.}}
```

Then I edited the file in order to

- remove the backslash(es) in the middle of the word part
- adjust IR type for control sequences (0 into 2) such that the sorting et cetera went smooth.

As result I obtained 343 sorted and compressed index entries. For typesetting some back substitutions had to be made

- in between backslashes inserted again
- the substitution/insertion of special control sequences.

The total sorting time on my Mac Classic II was roughly 20 minutes, with 12200 IR comparisons, and 54149 words of memory used.[23] As format I used blue.tex.[24]

The user's guide for BLue's Format system takes some 400 IRs. The processing of the index takes 1 hour on my Mac, and a little over 1000 save stack positions.

A final test was about copy with font changes and accents as part of the IRs. This file has been supplied in the second example of this article.

13 Robustness

The weak point in this automated complicated process is the specification of the IRs. When wrong ones are supplied low-level TEX error messages will emerge, and definitely will put off a user. My only remedy to this is

- don't hardly use markup within the IR
- don't use active charcters as part of the IR
- don't use TEX special characters, especially backslashes apart from a single control sequence (a type 2 IR.)

I also print in the log file the number of elements to be sorted: n. This must be greater than zero. When n=0 is printed, the file of raw index entries is apparently empty, and 9 out of the 10 cases the asynchronous behaviour of the OTR is the cause, meaning that the file was already closed before it was written.

When the file is empty, as argument of `\filetoarray`, a message is delivered in the log file.

In the macros I have left some `\immediate\write`s, preceded by %, which can be used to follow what is compared. This is handy when spotting an out of order IR.

14 Looking backward

It all started with sorting an array in SiB. In the early stage of this paper I adhered the model to allow writing IRs to a file and also, as option, to write directly to the array. Because of the latter, I had to modify the OTR such that the array elements were extended with page numbers. I used the mark mechanism and it worked.[25] I abandonned this option for simplicity reasons. When restricted to writing to a file a reader does not have to bother about the OTR, and no redefinitions of the array elements are needed.

A price I had to pay for flexibility is the generation of the replacement texts for control sequences to be ignored, each time in every invocation of `\makexref`.
Another 'inefficiency' is the 'generation' of the ordering table.
Perhaps, I should just have included the 'tables' as such instead.
A final point is the general conflict with control sequences already in use.

[23] A dummy job which just stores the IRs but does not sort nor reduce the number of entries needed 34884 words of memory.

[24] While proofing this work, it needs less hand work, because the control sequences can be accounted for via sortkey pairs. See the Looking forward section.

[25] However, there was still a detail to be fixed: the first mark was also written in the main vertical list.

15 Looking forward

Is it realistic to expect that there will be another user besides me? I don't think so, because history has it that manmac as such has been neglected at large, and I don't see why people would nevertheless prefer my work, which is so intimately connected to manmac.
Whatever the future has in store, let us go on.

15.1 Subsidiary entries

What are the requirements for this? Let us assume that the markup for a subsidiary entry starts with `\sub`. IMHO the following specs are relevant

- `\sub` should be neglected in the copy
- write `\sub` as string to the file index
- neglect `\sub` while sorting
- 'reduce' entries which differ in subentries by suppressing the main entry except for the first time, while typesetting the index element
- indent the subsidiary entry by `\parindent` while typesetting.

The above specs can be realized as follows

- `\let\sub=\relax`
- include `\sub` in `\conseqs`
- the reduce problem is similar to the typesetting of references, where the author part is printed once for different publications. I found that in $\mathcal{AMS}$-TEX, and used it in 'BLUe's References.'
- add `\let\sub=\endgraf` in `\pasteupreferences`.

Actually, the above has been incorporated in blue.tex.

15.2 Sorting keys

So far we have mostly neglected tokens while sorting. But, . . . is it do-able to sort each control sequence according to a sorting key?

This comes down to

- provide a way to specify for sorting key pairs, that is pairs of a control sequence and its sorting key
- sort on the sorting key
- set the index entries with the embedded control sequences according to their definitions.

Example *(Use of sorting keys)*

Suppose that we have

```
\add\fourtex{4tex}to\srtkeypairs
\add\fourtex to\srtkeys
%or \setupsrtkeys
Copy with ^{IR \fourtex}
%
\sortindex %with 4tex for \fourtex
\pasteindex%Set 'IR \fourtex{} <pagenumbers>'
\bye
```

then the file index will contain the IR

`IR \fourtex{} !0` ⟨*pageno*⟩

The above index element will be sorted on 4TEX. The key issue in the implementation is to extend `\nxtwindex` by the test `\pophead` $\in$ `\srtkeys`. If the test yields true sort further on `<sorting key><tail>`, that is insert the sorting key. Actually the above has been included in blue.tex with as defaults

```
\conseqs{\c\space\bf\it\rm\tt\sub\relax}
\consyms{\'\'\"\^\~}
\srtkeypairs{\TeX{tex}
             \LaTeX{latex}
             \AmSTeX{amstex}}
\srtkeys{\TeX\LaTeX\AmSTeX}
```

In order to use this nice feature extend the script as follows

```
\add...to\srtkeypairs%one or more pairs
\setupsrtkeys        %extends the set of
                     %sort keys
...%copy proper
\sortindex
\pasteupindex
```

The suppression of the word of the main entry with multiple sub entries can be implemented too. For the moment I refrained and wait for user response.

16 Availability

The macros are part of BLUe's Format System, and stored in tools.dat. The system is released on NTG's 4(All)TEX CD-ROM and the CTANs.

Used from manmac are the IR macros, the circumflex ^ and its auxiliaries, next to some macros for handling the doublecolumns, for example `\balancecolumns`. Some macros from SiB are used too, especially the sorting macros.

17 Acknowledgements

Erik Frambach thank you for the file of 4TEX, your assitance, and your sound judgement. As usual Jos Winnink helped me again to procrust the markup of the article into `maps.sty`.

18 Conclusion

The macros work smoothly for a modest and practical index. I have used the macros for the user's guide of BLUe's Format system 'Publishing with TEX.' The macros serve an educational purpose, and stimulate thinking.

My added value to manmac is, that a modest index can be processed in a one-pass job. Accents, to be ignored tokens, and sorting keys can be supplied. More complex indexes can be processed via a proof run, editing of index.elm, and a final run with index.elm inserted as copy.

For a foolproof approach it is necessary to look ahead for general tokens. The problem is that we can't account for all the control sequences or active characters to be used. For indexes of modest complexity my limited approach is good enough. The bonus is simplicity throughout.

In general it is difficult to know when to stop, as Schumacher in his precious 'Small is beautiful' already pointed out. My case rest.

References

I borrowed ideas undoubtedly from all the material I read. Most of the literature I read is contained in my literature database lit.dat. To my knowledge no-one handled the preparation of an index completely within TEX, before.

BLUe's Letters

With love

Kees van der Laan

Hunzeweg 57,
9893 PB Garnwerd, The Netherlands
cgl@rc.service.rug.nl

Abstract

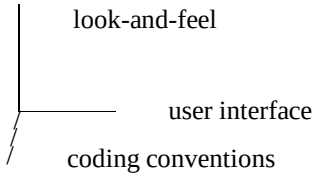
The backgrounds, use, design, and coding of BLUe's Letters format have been discussed. The purpose is to format a letter, merge it with address(es) from a database, and typeset it all with the appropriate background such as logo and the like, completely within TEX. Separate labels can be obtained too, either specified by name or searched for by pattern.

Keywords: Active list separator, addresses, address labels, compatible extension, databases, data integrity, education, fifo, lazy evaluation, letter, list element tag, macro writing, mail-merge, no-nonsense, pattern matching, plain TEX, reusable software parts, set macros, software engineering.

1 Introduction

What makes a letter different from a note? From an article? A key issue is the merging of the background (logo and the like), with address(es) from a database, and the letter proper. Of course the look-and-feel is much different too, although scientific communication was done by letter in the old days, instead of via articles in journals for an anonymous audience. This sets the scene when we like TEX to take care of it all.[1]

When designing a format three issues have to be dealt with, preferably as a mutual orthogonal set



In the sequel I'll talk about the

- use (and customization)
- look-and-feel
- user interface, and
- coding.

The `\letter` format is part of BLUe's format system, which is a personalized format and compatible with plain TEX.

I started from Knuth's (example) letter format, *The TEXbook* Appendix E. Knuth could do with the following handful of tags

```
\letterhead \address \body \closing \endletter
```

If this is such a great approach, how come that it has not been taken over? IMHO, with all respect, it was too difficult to customize. Moreover, at that time many a TEXie was busy porting the system to the various platforms. Nowadays I would like to

- store addresses in a database, and use these from there
- have a simpler letterhead
- let `\headline` take care of the headers
- use fewer fonts
- use the known sender — remember BLUe's format is yours, it knows about your context
- use window envelopes
- introduce token variables for `\subject`, `\yourreference`, and `\ourreference`
- make positioning of the address flexible, such that it can move to adjust to your window envelope
- adapt `\makelabel`[2] to work independenly with the (address.dat) database (Address labels are handy for carbon copies among other things)
- substitute `\beginscript` for `\beginletter` and ditto for the `\endscript`.[3]

2 Notations and definitions

`\ea` denotes `\expandafter`. `\nx` denotes `\noexpand`.

An important notion is what I call 'list element tag.' This element is a chameleon and takes any actual meaning to process the list elements. It is very close to the concept

[1] The advantage of handling all by TEX is that we don't have to bother with jargon from other tools. Another advantage is the stability, because TEX has been frozen.

[2] I named it `\makelabels`.

[3] Well, I decided to allow for the aliases `\beginletter` and `\endletter` too.

of active list separator. Knuth used this already[4] in *The TEXbook* Appendix D.2 List macros, the \\s, with the annotation 'But in fact, the \\ separators are enormously useful, because we can define \\ to be any desired one-argument macro, and then we can execute the list!' I prefer the name list element tag instead of (active) separator, because it does not really separate — the first element is different — moreover, the idea is not restricted to one argument. I have used the list element tag with name \lst. Agreed, to use the list element tag is unusual and confusing at first, but once you get the hang of it, you will appreciate it. It is an eye-opener.[5] Knuth has used the list element tag, \ansno, for formatting the answers of the exercises for *The TEXbook*. \ansno takes two arguments ended by period . and colon : respectively, and followed by the answer proper, typeset on the fly.

3 Use

The scripts for typesetting a letter vary with whether a typed-in letter or a stored letter will be merged. It also depends on the number of addressees. Extremes are a letter addressed to one person, or a letter sent to the complete database.

3.1 Typesetting letters

Below push-the-button scripts have been supplied by example. I assume that the addresses are available in the address.dat database,[6] as a list of elements, each obeying the syntax: list element tag, \lst, followed by a name tag, and a group. For addresses this comes down to the following.

```
\lst\<name>{<salutename>
\\<fullname>
\\<affiliation>
\email{...}
\phone{...}
\fax{...}  }
```

For <tag> I take the name with initials. <salutename> is the name to be used after 'Dear' or so. <fullname> is the name as it will appear on the envelope, and in the headline of the follow pages. The affiliation can be split over several lines — visual markup — but do precede each line by \\ — a lower level list element tag :-) — to denote for the time being a new line in the result. I adopted the convention to supply country names in capitals. Consistency facilitates the use of the \search macro, to match patterns.

Example *(Typeset a typed-in letter)*

```
\input blue.tex    %Personalized format
\letter            %knows logo, sender etc.
\subject{...}
\ourreference{...} %just numbers
\yourreference{...}
\addresses{\<name>}%to load address(es)
\addressee{\<name>}
\beginscript
\dear              %uses \addresseename
....
\sincerely         %knows about you
\cc ...            %set \item{cc.} ...
\ps ...            %set \item{P.S.} ...
\appendix{<appendixtitle>}
<appendixmaterial>
\endscript
```

The \appendix control sequence takes one argument. It starts a new page, but continues page numbering. \beginletter is an alias for \beginscript, analogous for \endletter.

Example *(Typeset a stored letter)*

If the letter proper is stored in letter.tex, then the outer-level markup looks even simpler.

```
\input blue.tex \letter
\subject{...}
\ourreference{...} %just numbers
\yourreference{...}
\letterto{\<name>...}
```

If we wish to send the letter to the complete database replace the last line by \lettertoall.

3.2 Making address labels

Although the addresses are integrated with the letter and suitable positioned for window envelopes, the separate address lables can be handy for carbon copies.

Example *(Some or all labels)*

```
\input blue.tex \letter
\makelabels{\knuthde\ntg}
\bye
```

All address labels from the database, specified in token variable \addressfile, can be obtained via

```
\input blue.tex \letter
\makealllabels
\bye
```

Labels which match a pattern emerge after

```
\input blue.tex \letter
\search{RUSSIA}
\makesearchlabels
\bye
```

[4] Of course. . . .

[5] The history is that I became familiar with the list element tag when working on the Tower of Hanoi in TEX. Since then I also used the fifo paradigm and could get rid of the list element tag at the expense of recursion. Now I combine the best of both worlds. The user only has to specify the list of names, and when reuse is in sight the macros fill the token variable \namelst with the names and the list element tags inserted, to facilitate later processing. When I started to work on handling references I first provided a list of definitions and loaded all those definitions from file. At that time TEX definitions were appropriate as entries. Only when I considered to load selectively, I found to use the \def as list element tag. Finally, I realized that the \def is confusing and dangerous and returned to the idea of lists preceded by a less vulnerable list element tag. The double backslash is so heavily used for e-o-l that I refrained from following Knuth on that. Furthermore, I found it convenient to associate a name to each list element.

[6] The file \addressesfile contains the name of the address database.

3.3 Extending the address.dat database

Example *(Entry of address.dat database)*

```
\lst\ntg{NTG

\\Nederlandstalige \TeX{} Gebruikersgroep
\\Postbus 394
\\1740 AJ Schagen
\\The Netherlands
\email{ntg@nic.surfnet.nl}
\phone{}
}
```

To add an address just adhere to the syntax, and include the new address in address.dat, in the right order, i.e., alphabetically sorted.

Check for data integrity

In order to make sure TEX can scan your address.dat, after you have added addresses, make a table of contents as follows.

Example *(Table of contents of address.dat)*

```
\input blue.tex

\contentsdatabase{address}
\bye
```

As a result there are no pages of output (that is OK!), and the file contentsaddress has been made. Note that I have already included `\newwrite \tocaddress` in blue.tex.

4 Customization

When starting to use BLUe's format system a user is asked to personalize the format. Name and affiliation have been customized, of course. For letters the logo is relevant, next to the bank account or so which can be supplied in the toks variable `\businessaccount`.

```
\businessaccount{Giro: C.G. van der Laan
   no {\oldstyle1321224}}
%
\def\logo{\copy\ntglogobox}
%
\def\ntglogo{\vbox{%
  \hbox{{\calx N}ederlandstalige}
  \hbox{\hskip1em{\calx T}\kern-.2ex
        \raise-.5ex\hbox{E}\kern.1exX}
  \hbox{\hskip2em{\calx G}ebruikersgroep}}}
\setbox\ntglogobox\ntglogo
```

The skip variables `\haoffset`, and `\vaoffset` — mnemonics horizontal address offset, vertical address offset — can be used to adjust the address to the window of your window envelopes.

To allow for your address database, say `<name>.dat`, include

```
\addressesfile{<name>}
\newtoks\toc<name>
\immediate\openout\toc<name>=contents<name>
```

If your file which contains the letter is different from letter.tex provide `\letterfile{<filewithletter>}`.

Customization towards a different design is beyond the scope of this paper.

5 Look-and-feel

A letter is characterized by a first page with a special header part, an address window, letter beginning, and footer. On follow pages it must be clear which first page each follow page is supposed to follow.

5.1 Letter

The following is a sketch of the design. Just the rough outlines.

I decided to set a logo left upper and sender affiliation right upper. Then follows, separated by a line, reserved space — let us call this a window — to typeset the address in. This is followed by reference information left and date right. The letter opening starts with a salute followed by the letter proper, possibly split over several pages. At the end the signature of the sender is set right. The back matter consists of P.S., cc., or Appendix with a title. The first page footline is separated by a partial horizontal rule below which the business information has been set.

A follow page takes a modest headline with addressee left upper and subject, our reference, and date right upper. The footline contains just the page number.

5.2 Address label

Each label is set with the address in the middle and the sender affiliation left at the bottom of a window of 6×13 cm.[7]

6 User interface

How should the markup language for the user look like? In agreement with BLUe's format the letter is a script with info elements preceding the script proper. I consider addresses also information, similar to the references of an article. The addresses can be loaded selectively from the database via

```
\beginaddresses
\<name>...        or   \addresses{\<name>...}
\endaddress
```

The use of the `\script` tags is in agreement with blue.tex.

6.1 Letter

The information tags appear before `\beginscript`, and can be looked upon as token variables.[8]

```
\addresses{...}   %load list of addresses
\subject{...}
\ourreference{...}%digits because I fancy
                  %frisky \oldstyle
\yourreference{...}
```

The blue collar workers read

[7] Well it depends. `\twocol` will yield smaller labels.

[8] They can be supplied in any order. This a general aspect of BLUe's format system.

```
\addressee{...}   %splits off name
\beginscript...\endscript
```

In the letter proper ordinary TEX markup can be used, next to the definitions[9]

```
\dear
\sincerely
```

The back matter has the markup tags

```
\ps
\cc
\appendix{...}
```

Special cases are handled by

```
\letterto{...}    %list of name tags
\lettertoall
```

6.2 Address label

The markup tags read

```
\makelabels{...}  %name tags
%
\search{...}
\makesearchlabels
%
\makealllabels
```

7 Coding

In BLUe's format system two-part macros are the starting point. A one-part macro is provided on top of it. Specification of information and actual typesetting have been separated.

7.1 Handling addresses

In `\beginaddresses` the names are first defined and then overloaded by their entry from address.dat, the (default) file specified in `\addressesfile`. Also the token variable `\namelst` will contain the list of the names of the loaded elements each preceded by the list element tag, `\lst`, to facilitate execution of the list.

```
\def\beginaddresses#1\endaddresses{%
  \def\process##1{\ifx\undefined##1
    \let##1\addresserror\else
    \message{\Dash\string##1
         already loaded.\Dash}\fi
  \namelst\ea{\the\namelst\lst##1}}
  \fifo#1\ofif%end defining all names
  \loadselectivefrom{\the\addressesfile}}
%with on top the trivial variant
\def\addresses#1{\beginaddresses
                 #1\endaddresses}
%and auxiliary
\def\addresserror{Address not in databasea
    (Sorry).\loaderror{Addresses}}
```

`\loadselectivefrom` reads as follows.

```
\def\loadselectivefrom#1{%#1 address or lit
 \def\lst##1{\ifx##1\undefined\ea\gobble
    \else\ea\gdef\ea##1\fi}\input #1.dat}
```

7.2 The markup for a letter

We have two aspects the user macros, and the page makeup.

User macros

`\letterto` executes `\processletter` for each address supplied as argument. `\processletter` provides the script and takes the information it needs. It also splits the address. Note that the `\letterto` was needed to alleviate for the user the trouble to retain values for the next letter, to be processed in the same run. This is induced by my use of `\headline`, which is changed for follow pages.

`\lettertoall` sends the letter to all addresses from the address database as specified in `\addressesfile`. For all those addresses `\processletter` is executed.

`\addressee` takes the first two lines apart from the address entry for `<salutename>` and `<fullname>`, and sets the address in the affiliation box.

```
\def\letterto#1{{\everyscript{\notlastscript}
   \storedvsize\vsize
   \storedheadline\headline
   \storedfootline\footline
   \beginaddresses#1\endaddresses
   \let\lst\processletter\the\namelst}}
%
\def\lettertoall{{\everyscript{\notlastscript}
   \def\lst##1{\processletter}%pick address
   \input\the\addressesfile.dat\relax}}
%
\def\addressee#1{\ea\splitaddress#1\egroup}
%
%with at the lower level
%
\def\processletter#1{{%#1 name or address
   \headline\storedheadline
   \footline\storedfootline
   \vsize\storedvsize
   \addressee{#1}
   \beginscript
   \input\the\letterfile\relax
   \endscript}}
%
\def\splitaddress#1\\#2\\{\addresseename{#1
   \unskip}\fullname{#2\unskip}\setbox
   \affiliationbox=\hbox\bgroup#2\\}
```

Page makeup

usually implies the output routine, OTR for short. However, for this application I could do without modifying the OTR.

In contrast to the letter format as supplied in Appendix E of *The TEXbook*, I heavily used `\headline` and `\footline`. The big deal is the headline. It consists of a vbox of height 3cm, with logo and sender affiliation, followed by an `\hrule`, and the address window. Note that `\headline` is redefined for use on subsequent pages. This implies that the original `\headline` has to be stored for the next letter. This holds too for `\footline`. Note that the `\vsize` for follow pages is changed in `\footline`.

[9] I don't know of a trick to remember that `\dear` is a definition and not a token variable to represent the language dependent word 'Dear.' Perhaps it is good to remind that the letter format knows about the addressee, once the name of the address has been supplied.

```
\headline{\line{\vbox to3cm{%
\line{\logo\hss
  \vbox{\hsize.33\hsize\small
   \the\address\\\the\netaddress}
}\kern3pt\hrule\vss}\hss}%
\addresseewindow
%and for follow pages
\global\headline{\line{\vbox
  to3cm{\vss%Implicit vspace
  \line{\tenrm To: \the\fullname
   \hss\today/\the\subject/\the\crowner\
   \oldstyle\the\ourreference}\kern2pt\hrule
   \vss}}}%end follow \headline
}%end \headline
%
\footline{\line{\vbox{%
  \kern\baselineskip\hrule\kern.5ex
  \hbox{\strut\the\businessaccount}}\hss}
%and for follow pages
  \global\vsize19cm
  \global\footline{\line{\null
   \hss--{\oldstyle\the\count0}--\hss
   }}%end follow \footline
}%end \footline
```

The `\addresseewindow` sets the address in a window of 4cm height and width `\hsize`. The positioning is biased by the values of the skip variables `\vaoffset`, and `\haoffset`.[10] This is followed by the information elements.

```
\def\addresseewindow{\line{%
\vbox to 4cm{%    %Window height Dutch
                  %envelopes
\vskip\vaoffset   %To shift address vert.
\leftskip\haoffset%To shift address hor.
\unhbox\affiliationbox
\vss}\hss}%end line
\line{\hbox to\longindentation
   {\hbox to8ex{Subject\hss}:
     \the\subject\hss}\today\hss}
\line{\hbox to8ex{\small Our Ref\hss}:
   \the\crowner\
   \oldstyle\the\ourreference\hss}
\line{\hbox to8ex{\small Your Ref\hss}:
   \the\yourreference\hss}
}
```

For completeness the following

```
\def\beginscript{\lastscript
   \the\everyscript\the\thisscript
\begingroup\pageno1 \null
\vskip3\bigskipamount
}%end \beginscript
%alias
\let\beginletter\beginscript
%
\def\endscript{\smallskip
  \vfil\eject\endgroup
  \tracingstats1
  \stop\thisscript{}}
%alias
\let\endletter\endscript
%
\def\dear{Dear \the\addresseename,\bigskip}
%
%To be replaced by your salutation
\def\sincerely{{\bigskip
   \parindent\longindentation
   Sincerely,
   \medskip
   \the\author\vskip3\bigskipamount}}
%
\def\ps{\bigskip\small\item{P.S.}}
\def\cc{\bigskip\small\item{cc.}}
\def\appendix#1{\newpage\tenpoint
   \centerline{\bf\the\appendixname\ #1}
   \bigskip}
```

Defaults are

```
\onecol%Because in blue.tex \twocol default
\addressesfile{address}
\searchfile{address}
\letterfile{letter}
\def\email#1{}\def\phone#1{}
%Separation headline and rest
\vsize13cm%First page
\hsize13cm\pagewd\hsize
\hoffset1cm
\parindent0pt
\generalindent2pc
\interlinepenalty1000
\longindentation.667\hsize
\storedvsize\vsize
\storedheadline\headline
\storedfootline\footline
\raggedbottom
```

7.3 The markup for an address label

Address labels can be obtained by `\makelabels` with the names as arguments. Another possibilty is to use `\search` with a pattern as argument — which will yield the address names, each preceded by `\lst`, in `\namelst`, and defines the names with associated list element as replacement text — together with `\makesearchlabels`. The last possibility is to use `\makealllabels`.

```
\def\makelabels#1{\vsize=28cm%
   \headline{}\footline{}%
   \beginaddresses#1\endaddresses
   \let\lst\processlabel\the\namelst}
%
\def\makesearchlabels{\vsize28cm%
   \headline{}\footline{}%
   \let\lst\processlabel\the\namelst}
%
\def\makealllabels{\vsize28cm%
   \headline{}\footline{}%
   \def\lst##1{\processlabel}%
   \input\the\addressesfile.dat\relax
}
%
%with at the lower level
%
\def\processlabel#1{\addressee{#1}%
   \boxit{\kern1cm\vbox to3.5cm
      {\noindent\leftskip.33\hsize
       \hsize.9\hsize
       \unhbox\affiliationbox\vss}
   {\smallskip\small
        \leftskip\generalindent
         \the\author\\
         \the\address\bigskip}
}\smallskip}%end \processlabel%;nonum
```

[10] Note the subtle use of `\unhbox` for the affiliation box.

7.4 Table of contents of database

A stepping stone application of Knuth's list element tag, see *The TEXbook* Appendix D, is provided in the macro below to yield a list of all the names of the entries in a database.

```
\def\contentsdatabase#1{%#1 pic lit address
\ea\let\ea\name\csname toc#1\endcsname
\immediate\openout\name=contents#1
\def\lst##1##2{\immediate\write\name{\nx##1}}
\input #1.dat\relax}
%with auxiliary
\newwrite\toc<#1>
```

Explanation. The list element tag — for these class I databases `\lst` — gets the meaning to extract the `\<name>` of each list entry, and store these in the file `contents<#1>`.

7.5 Search by pattern

The idea is that entries from the database specified in `\addressesfile` can be located via searching for a pattern. Of the found entries the names are collected in the token variable `\namelst`, with each name preceded by `\lst` to facilitate processing later. Moreover the names are defined with the associated list element proper as replacement text, i.e., the spotted entries are loaded.

```
\def\search#1{\def\loc##1##2{%
   \def\locate####1##1####2\end
    {\ifx\empty####2\empty\foundfalse
    \else\foundtrue\fi}\ea\locate##2.##1\end}
   \def\lst##1##2{\loc{#1}{##2}\iffound
     \immediate\write16{\nx##1}%log file
     \namelst\ea{\the\namelst\lst##1}
     \def##1{##2}%define found element
    \fi
   }\input\the\searchfile.dat\relax}
```

8 Test program

The following is in use by me to test the `\letter` format. When you change some parts insert the changed macros after the first line. I assume that a prototype letter has been stored in letter.tex.

```
\input blue \letter
\letterto{\knuthde\grinevaoa}
\bye
\lettertoall
\bye
\makelabels{\knuthde\grinevaoa}
\bye
\makealllabels
\bye
\searchfile{address}
\search{RUSSIA}
\makesearchlabels
\bye
\contentsdatabase{address}
\bye
```

9 Summary of tags

Personalize these tags, make it your crib. The numbers refer to the page numbers in fmt.dat — BLUe's format database of formats — when printed by pgfile.tex, or via the script as provided in the beginning of fmt.dat.

Token variables already available in blue.tex are

- `\vaoffset` and `\haoffset`, to position the address
- `\letterfile`, which contains the name of the file of the letter, and
- `\addressesfile`, which contains the name of the address database.

```
%Address
%  \beginaddresses..........9-19
%  \endaddresses............21
%  \addresses..............23-24
%Letter
%  \addressee..............26-27
%  \letterto..............101-107
%  \lettertoall...........109-110
%lower level
%  \splitaddress............51-54
%  \processletter.........151-158
%Header and footer
%  \headline..............201-214
%  \footline..............216-224
%lower level
%  \addresseewindow.......251-266
%  \vaoffset, \haoffset...254-254
%Labels
%  \makelabels............301-304
%  \makealllabels.........306-309
%  \makesearchlabels......311-313
%lower level
%  \processlabel..........351-360
%Composition
%  \beginscript...........401-406
%  (\beginletter).............407
%  \endscript.............409-412
%  (\endletter)...............413
%  \dear......................508
%  \sincerely.............502-506
%  \ps........................510
%  \cc........................511
%  \appendix..............512-514
```

10 Acknowledgements

Many a suggestion with respect to simple markup tags in the user interface was done by Erik Frambach. Discussions with Herman Haverkort on the TEX-NL network contributed to a clearer notion of the list element tag. Jos Winnink proofed as usual. Thank you!

11 Conclusion

More or less to complete BLUe's format the `\letter` format emerged. While working on it, it was fun to experience that the database mechanisms as developed for references could be reused in this context.

References

The TEXbook and LATEX user's guide are omni-present and not explicitly listed. For my works consult lit.dat, via the use of `\search` for example.

BLUe's Reports

Good Work

Kees van der Laan

Hunzeweg 57,
9893 PB Garnwerd, The Netherlands
cgl@rc.service.rug.nl

Abstract

The backgrounds, use, design, and coding of BLUe's Reports format have been discussed. The purpose is to format a report as a compatible extension of the default note format of BLUe's format. As such it is an example of how to customize BLUe's format system with a format of your own. The differences have to do with
not only using titles but also reusing the titles for table of contents, list of examples, and running headlines
the preliminary pages and cover pages which are absent with the default note format
appendices may contain collected information from the script such as answer to the exercises (similar as in *The TEXbook*), table of examples, table of contents, and an index.

Keywords: Active list separators, compatible extension, databases, data integrity, education, fifo, house style, lazy evaluation, list element tag, macro writing, no-nonsense, on the fly, pattern matching, plain TEX, reports, reusable software parts, separation of concerns, set macros, software engineering.

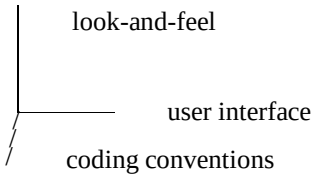
1 Introduction

What makes a report different from a note, form an article? What is the added value? BLUe's format default note style allows `\onecol`, but there is a lot more to be added if we really want to format a script in such a way that a report will emerge. The following list summarizes the differences.

- The front matter is more elaborate (preface page, title page, abstract page, foreword page, contents page).
- The copy proper is divided into chapters, with the tree of sections and subsections underneath, and appendices as special chapters. The latter contain for example dynamically generated tables of contents, examples, figures, etc, and an index. Also answers to exercises can be collected in an appendix.
- The cover matter (front, back and the inside covers).

This sets the scene when we like TEX to take care of it all.[1]

When designing a format the following three issues have to be dealt with, preferably as a mutual orthogonal set.



In the sequel I'll talk about the

- use (and customization)
- look-and-feel
- user interface, and
- coding.

For the `\report` format the new aspect is the look-and-feel of a report, because the coding conventions have already been decided upon in BLUe's format, ditto for the user-interface philosophy.

The more complex codings are a result from the

- two-part macros as basis
- one-part macros on top
- storing of the 'arguments' with the right catcodes, and
- writing of titles to a file with page numbers attached.

The creation of an index in one-pas is not special for `\report` and has been treated in 'BLUe's Index.' The use of `\sortindex` and `\pasteupindex` has been explained in 'Publishing with TEX,' the user's guide for BLUe's format system.

1.1 Notations and definitions

`\ea` denotes `\expandafter`. `\nx` denotes `\noexpand`.

[1] The advantage of handling all by TEX is that we don't have to bother with jargon from other tools. Another advantage is the stability, because TEX has been frozen.

[2] Of course. . . .

An important notion is what I call 'list element tag.' This element is a kameleon and takes any actual meaning when processing the list elements. It comes close to the concept of an active list separator. Knuth used this mechanism already[2] in *The TEXbook* Appendix D.2 List macros, the \\s, and stressed the potential with the following remark. 'But in fact, the \\ separators are enormously useful, because we can define \\ to be any desired one-argument macro, and then we can execute the list!' I prefer the name list element tag instead of (active) separator, because it does not really separate — the first element is different. Note that the idea is not restricted to one argument. The list element tag has been applied in for example `\contentsdatabase` with the list element tag `\def`. Agreed, the use of a list element tag is unusual, but once you get the hang of it, you can read through it, and you will recognize the essential issues.[3] Knuth used the list element tag, `\ansno`, for formatting the answers of the exercises for *The TEXbook*. `\ansno` takes two arguments ended by period . and colon : respectively, and followed by the answer proper, typeset on the fly.

2 Use

First a disclaimer. A report format is only handy when there are chapters, and when we like to see the preliminary pages in print as well. If not, stay with the default note format and switch to `\onecol`. This prompts for proofing via the default format.

3 The markup of a report

Example *(Just a chapter)*

To typeset a chapter, forget about `\beginscript`, which typesets the cover, and the preliminary pages.[4]

```
\input blue.tex \report
\chapterhead{...}
<copyproper>
\bye
```

Example *(Template for a report script)*

```
\input blue.tex
%\loadindexmacros%When creating an index
\report          %Preliminary matter follows
\title{...}
\subtitle{...}
\keywords{...}
\abstract{...}
\bibliographydata{ISBN: ...}
\acknowledgements{...}
\pictures{...}
\preface{...}
\begincontents%\se is subentry descriptive tag
  Preface
  Table of contents
  Introduction
  \se BLUe's format
  ...
  \se J: Table of Contents
\endcontents
%\let\coverpic=\<name>pic
\beginscript
\input pwt.int   %Introduction
...
\setupappendices %Appendices A, B, C,...
\pasteupanswers  %Answer to Exercises
...
\pasteuptoe      %Table of Examples
%\sortindex\pasteupindex
\pasteuptoc      %Table of Contents
\endscript
```

In the format the necesssary files have been opened via

```
\immediate\openout\<name>=<filename>
```

with for example `<name>` = ans, toc, toe, and `<filename>` = answers, contents, and examples, respectively.

3.1 Proofing a single chapter

Chapters can be proofed separately as follows.

```
\input blue.tex       %\loadindexmacros
\report               %After \loadindexmacros
\input <filewithchapter>%\sortindex\pasteupindex
\bye
```

We can easily test a few chapters in a row as follows. Omit `\bye` and when TEX prompts for an `\end` insert 'i' for insert and after the insert prompt supply `\input <nextchapter>`. The testing of index entries per chapter on the fly is convenient.

4 Customization

Remember that BLUe's format system is a personalized format, it knows about you. Candidates for customization are the preliminary pages and the back cover. For the front cover a picture can be assigned to `\coverpic`. The redesign of `\beginscript` and `\endscript` is beyond the scope of this paper.

5 Look-and-feel

The design in broad terms yields a cover page (with title matter and cover picture), an inside cover (with keywords, bibliography information, acknowledgements), a title page, (a translator page if relevant), a preface page, and a contents page. This static contents page can assist the writing as such, to maintain the survey. `\report` also generates a table of contents on the fly with page numbers attached.[5]

[3] The history is that I started with loading all the references from my literature database. At that time TEX definitions were appropriate as entries. Only when I considered loading selectively from the database, I found using the `\def` as list element tag. Later I realized the vulnerability of `\def` especially when the OTR can come in between. The
is heavily used for e-o-l so I can't follow Knuth in that. Now I use `\lst`.

[4] Ditto for `\endscript`, which takes care of the back cover.

[5] If you don't like this static contents page idea, let `\pasteupcontents` equal `\relax`.

Details of the layout are treated along with the coding.

6 User interface

The approach initiated in BLUe's format of providing information via token variables[6] in the preliminary part has been followed. The `\beginscript` typesets the preliminary pages.

The appendices are set via `\pasteup<name>` now and then, to abstract from details. `\setupappendices` provides for letters as numbering.

Another problem I had to face was how to cope with names. In BLUe's format I used as principle a 'root with affixes.' I kept to that approach although not unduly. I tried to remain consistent but at the user level the names should not be too long. At the lower level it is nice to have long and systematic names, which prevents unintended double meanings.

7 Coding

In BLUe's format two-part macros are the starting point. A one-part macro is provided on top of it. The aim is to allow processing on the fly.[7] Specification of information and actual typesetting have been separated for matter which does not belong to the copy proper, such as abstract, acknowledgements, keywords, preface and the like.

7.1 New control sequences

Examples of new control sequences are for example those associated with `\preface` and `\chapterhead`.

Preface matters

The preface can be specified via

```
\beginpreface
...                 or      \preface{...}
\endpreface
```

The coding reads

```
\def\beginpreface{\setbox\prefacebox
   \vbox\bgroup}
%
\def\endpreface{\egroup}
%
\def\preface#{\beginpreface\bgroup
 \aftergroup\endpreface
 \afterassignment\ignorewhitespace
 \let\dummy=}
%
\def\pasteuppreface{\centerline{\chapterfont
   \the\prefacename}\bigskip
   \unvbox\prefacebox}
```

The one-part macro on top is apart from some checking and neglecting of white space, equivalent to

```
\def\preface{\aftergroup\endpreface
   \beginpreface}
```

Although the use via `\preface{...}` might suggest that catcodes will be assigned too early to the argument tokens, this is not the case, due to the special coding. To understand the coding these kinds of mechanisms must be known and recognized. The preface is pasted up in `\beginscript`.

Chapter head matters

The required functionalities — the specs so to say — are to

- start a new page
- store the title for reuse (running head, table of contents)
- maintain and store the chapter counter (for use with the exercises, table of contents, and the like)
- write the title and title counter value to the contents file
- typeset the title on a new page with some white space after
- reset counters which start anew per chapter (headings, exercise number, and the like)
- provide the right `\headline` and `\footline` with the title, chapter title, and page number
- suppress the headline on the first page of a chapter (or supply for an otherwise special headline)
- process the title on the fily to allow for example verbatim
- condition the writing of the title and title counter value to the examples file.

Quite something, isn't? That is why title matters are not trivial.[8] The conditioning of writing the title to the examples file is there, because we don't like the title etc. to appear in that file if the chapter lacks examples.

```
\def\beginchapterhead{\the\prechapterhead
  \storechaptertitle}
%
\def\storechaptertitle#1\endchapterhead{%
   \global\chaptertitle={#1}\endchapterhead}
%
\def\endchapterhead{\centerline{\chapterfont
   \the\chapternumbering\the\chaptertitle}
   \the\postchapterhead\ignorewhitespace}
```

This approach looks promising. What has to be done before is the concern of `\prechapterhead`. `\storechaptertitle` does what its name suggests. The formatting of the number of the chapter is the concern of `\chapternumbering`. `\postchapterhead` takes care of leftovers. Default `\chapternumbering` is empty, that is, no number will be typeset.

Undoubtedly, the careful reader would object that the title is stored and that the catcodes are set before processing. That is right, of course, but via `\everychapterhead` —

[6] A white lie, it looks like it. . . .

[7] When arguments are not processed on the fly the catcodes of the tokens of the argument are set while read by TEX's parameter scanning mechanism. That can be different from what is meant. An example is the circumflex in in-line math. In-line verbatim occurs much especially when TEXies write in TEX about TEX. Although I don't like use of different fonts in a heading, I nevertheless provided for the opportunity to do so.

[8] My 'Paradigm: Headache?' was complex enough, it paid attention to part of it, neglecting the storing and reusing aspects!

in \prechapterhead — we can guarantee that symbols will have the right catcodes. \storechaptertitle is there to allow the short variant \chapterhead{...}. The combination of storing and in-line verbatim does not work. Beware.[9]

```
\prechapterhead{\newpage\null\vskip4pc
   \bgroup\catcode'\^=7
   \the\everychapterhead\the\thischapterhead
   \global\advance\chaptercnt1 }
%
\postchapterhead{\egroup
   \headcnt0 \exercisecnt0 \examplecnt0
   \write\toc{\nx\separator
      {\alfanum\the\chaptercnt}
      {\the\chaptertitle}}
   %suppress headline first page chapter
   \headline{\global\headline
    {\hbox to\pagewd
     {\sl\the\chaptertitle\hfill
                   \the\title}}}%
   \footline{\hbox to\pagewd{\small
     \rlap{Draft \today}\hfill\dash
    {\oldstyle\the\pagenumber}\dash
     \hfill\llap{\copyright
                 \the\crowner}}}%
%Example is redefined here such that on
%first use if ever in the chapter, it
%redefines itself into the regular macro,
%while writing the chapter titles to the
%table-of-examples, ToE, file.
%In this way chapter titles of chapters
%without examples will not be written to
%the ToE.
   \def\example{\write\toe{\nx\separator
       {\alfanum\the\chaptercnt}
       {\the\chaptertitle}}
      \let\example\regularexample
      \example}
   \thischapterhead{}\vskip3pc}
```

The material is enclosed within a group to keep the effect of \thischapter local.

\example is defined such that only when it is used the title and number of the chapter will be written to the file associated with \toe. \regularexample writes the example title and number to the file associated with \toe.

Intriguing is \alfanum. The purpose is to control whether a number or a letter should appear. This is particularly useful with appendices.

```
\def\setupappendices{\thischapterhead=
   {\chaptercnt64 \gdef\alfanum{\char}%
    \global\chapternumbering=
    {\alfanum\the\chaptercnt: }}}
```

And what about the one-part macro on top? The above was already complicated. What is the problem for a one-part macro? For most of the cases we can do with a simple variant. However, when we like to process math, especially the circumflex, as part of the title more work has to be done. The mechanism has been explained in 'Paradigm: Two-part macros.'

```
\def\chapterhead{\bgroup
   \def\storechaptertitle##1{\egroup
     \global\chaptertitle{##1}%
     \endchapterhead}\beginchapterhead}
```

As a tribute to manmac I also defined the following limited variant.[10]

```
\def\bluechapter#1\par{\beginchapterhead
   #1\unskip\endchapterhead}
```

A trivial user interface on top which counteracts the curly braces mania at the expense of processing on the fly.[11]

7.2 Redefined control sequences

Important are the redefinitions of the various headers, and the \beginscript and \endscript macros. Details are the redefinitions induced by my making \parindent equal to zero, while I wish to retain indented \items and their derivatives. For that purpose I had to redefine \hang, \itemitem, and \textindent. I also had to adjust \exercise, \answer, and \ansno from manmac to this environment.

Pasteup matters

I do like these macroscopic control sequences very much. Their use is intuitive.

```
\def\pasteuptoc{%
 \chapterhead{Table of \the\contentsname}
\immediate\closeout\toc%Table Of Contents
{\ninepoint\def\contentsno##1.##2:
   {\noindent\hbox to\generalindent
   {{\oldstyle##1}.{\oldstyle##2}\hss}}
 \def\separator##1{\smallbreak\noindent
  \hbox to\generalindent{{\oldstyle##1}\hss}}
 \def\she{\noindent\hbox
        to1.5\generalindent{\hss}}
 \def\sshe{\noindent\hbox
        to2\generalindent{\hss}}
 \parindent0pc\obeylines\sl
 \input contents\relax}}
```

Note the use of control sequences for language-dependent names. The list element tag, \contentsno, gets its meaning, next to other quantities left undefined sofar. The contents proper is processed on the fly. Only the numbers are parsed as arguments.

Note that next to the dynamically generated ToC, there is the possibility to pasteup a static ToC, via \pasteupcontents. The control sequences \begincontents and \endcontents have been redefined in order to set the static contents via a \valign. No short variant is possible for these.

Header matters

Header macros have to be redefined because the header titles are also used in the the table of contents. In contrast with the default blue.tex the headers maintain a number, to

[9] For me this does not hinder because I don't like these kinds of font changes in a chapter header.

[10] The circumflex in math in the title will hiccup. Most of the time we don't need that.

[11] What does on-the-fly mean anyways, when we have to store the title for multiple use? The answer is that we would like to allow for math too. No verbatims however.

be used in the table of contents. Typical for the class of headers are the macros associated with \head. Because of the storing of the head title, the title is not processed on the fly. The one-part macro is not straightforward, because I allowed for adjusting catcodes via \everyhead.[12] This is handy when we like also math as part of the title.

```
\def\beginhead{\the\prehead
   \global\advance\headcnt1
   \storeheadtitle}
%
\def\storeheadtitle#1\endhead{%
   \global\headtitle={#1}\endhead}
%
\def\endhead{\centerline{\headfont
   \the\headtitle}\the\posthead}
%
\posthead{\egroup\xdef\writetoc{%
   \write\toc{\contentsno\nx\alfanum
    \the\chaptercnt.\the\headcnt:
    \the\headtitle\nx\nx\nx
    \pagenorepresentation{\nx\number
    \pageno}}}\writetoc
   \nobreak\medskip\noindent
   \ignorewhitespace}
%and adapted from blue.tex the default
\prehead{\vskip0pt plus9ex
   \penalty-250\vskip0pt plus-9ex
   \bgroup\catcode`\^=7
   \the\everyhead\the\thishead
   \bigskip\noindent}
```

\prehead is left invariant, that is, the discouraging of a page break after a heading has been left invariant, as the amount of white space before the heading. The catcode of the circumflex has regained its original value to allow for math as part of the title. The group with \egroup in \posthead is inserted to localize the effect of the catogory change of the circumflex, and the inserted values of \everyhead and \thishead. Note that \contentsno, the list element tag, will perform a similar function as Knuth's \ansno, namely, to process the list of elements contained in the file contents which is associated with \toc.[13] Also the markup for the formatting of the table of contents has been abstracted in descriptional markup. This is related to the concept of lazy evaluation.

Note too that no \immediate \write can be used, because of the asynchronism of the OTR. The page number expansion must also be delayed and done in the OTR.

The one-part macro reads as follows.

```
\def\head{\bgroup
  \def\storeheadtitle##1{\egroup
   \global\headtitle{##1}\endhead}\beginhead}
```

Script matters

\beginscript takes care of typesetting the preliminary pages, and \endscript's concern is to format the back cover. Within the approach taken the coding is not difficult, just tedious. I tended to ignore an element when not filled-in by a user.

```
\def\beginscript{\lastscript\begingroup
 \the\everyscript\the\thisscript
 \pageno-1 \headline{}\footline{}
\ifproof\rightline{\rlap{\sl Cover}}\fi
 \kern7\bigskipamount
 \centerline{\chapterfont\the\title}
 \bigskip
 \centerline{\subtitlefont\Dash
             \the\subtitle\Dash}
 \vskip\bigskipamount
 \centerline{by}
 \vskip\bigskipamount
 \centerline{\subtitlefont\the\author}
 \vfill
\ifx\coverpic\undefined
     \else$$\coverpic$$\fi
 \vfill
 \if!\the\translator!
 \else\rightline{\subtitlefont
   Translated by \the\translator}\fi
\newpage
\ifproof\rightline{\rlap{\sl Inside
                         Cover}}\fi
 \kern3\bigskipamount
 \pasteupkeywords
 \smallskip
 \the\bibliographydata
 \vfill\vfill
 \pasteupacknowledgements
\newpage
\ifproof\rightline{\rlap{\sl Title
                         Page}}\fi
 \kern7\bigskipamount
 \centerline{\chapterfont\the\title}
 \bigskip
 \centerline{\subtitlefont\Dash
             \the\subtitle\Dash}
 \vskip\bigskipamount
 \centerline{by}
 \vskip\bigskipamount
 \centerline{\subtitlefont\the\author}
 \vfill\pasteupabstract\vfill\vfill
\ifvoid\forewordtranslatorbox
 \else    \newpage
 \ifproof\rightline{\rlap
  {\sl Foreword Translation}}\fi
 \vfill
 \unvbox\forewordtranslatorbox
 \vfill
 \rightline{\vtop{\hbox{\the\translator}}
           \hbox{\the\author}}}\vfill
\fi
\ifvoid\prefacebox\else\newpage
 \ifproof\rightline{\rlap{\sl Preface
                         Page}}\fi
 \vfill\pasteuppreface\vfill\vfill
\fi
\newpage
\ifproof\rightline{\rlap{\sl Contents
                         Page}}\fi
\vfill\pasteupcontents
}%end \beginscript
%
\def\endscript{\xcol=\maxcols
   \vfill\eject\endgroup\backcover
   \tracingstats=1 \stop\thisscript{}}
```

\backcover is default \relax. For an example see \pwtbackcover in the fmt.dat file.

[12] It is analogous to the one part macro \chapterhead and \example.

[13] It is not expanded because at the moment of writing to a file it is equal to \relax.

Example matters

What are the required functionalities? I could think of the following.

- no page break immediate after the heading
- maintain an example counter
- write the counter value and the title of the example to the file examples which is associated with `\toe`, and attach the page number to it, for a proper list of examples[14]
- typeset the example name, and the title
- it should be possible to have in-line math as part of the title.

Together with `\report` I had to do something about the circumflex, because when expanded during the writing to the file it yielded an error messages. The easiest thing was to make it equal to `\relax` locally. However, what about the asynchronism of the OTR? No problem, because when it is read in `\pasteuptoe` it will be processed as in ordinary text. It is just the expansion while writing to a file which has to be suppressed. Despite that I chose at last for a local catcode change.

I could not provide for storing the title and also allowing for verbatims to be handled appropriately.

```
\def\beginexample{\vskip0pt plus5ex
 \penalty-100\vskip0pt plus-5ex
 \medskip
 \global\advance\examplecnt1
 \bgroup\catcode`\^=7 \the\everyexample
  \the\thisexample\storeexampletitle}
%
\def\storeexampletitle#1\endexample{%
  \global\exampletitle{#1}\endexample}
%
\def\endexample{%
  \xdef\writeexa{\write\toe{\exampleno
  \alfanum\the\chaptercnt.\the\examplecnt:
  \the\exampletitle
  \nx\nx\nx\pagenorepresentation
  {\the\count0}}}\writeexa
  {\the\examplename} {\sl(\the\exampletitle)}
  \egroup
  \nobreak\endgraf\noindent\ignorewhitespace}
```

For consistency reasons the following has been added.

```
\def\example{\bgroup
  \def\storeexampletitle##1{\egroup
   \global\exampletitle{##1}\endexample}
  \beginexample}
```

As a tribute to manmac the following limited variant has been added. No math with the circumflex is allowed.[15]

```
\def\blueexample#1\par{\example{#1\unskip}}
```

8 ToC tags

During this work I realized that many a tag should be included in the default blue.tex. An example of the latter is `\summary`. I also extended the font switching macros with `\large` and `\Large`. Both are also included in blue.tex. `\large` is used with transparencies, and `\Large` is in use for typesetting chapter headings.

For the list of tags I refer to 'Publishing with TEX.'

The macros are part of fmt.dat — BLUe's format database of formats — and can be listed with appropriate page numbers by gpfile.tex. A table of contents of the macros borrowed from fmt.dat has been included. Make your crib out of it.

```
%Table of Contents report format
%Token vars (declared in blue.tex)
%   \prefacename
%   \bibliographydata
%   \chaptertitle
%   \prechapterhead
%   \postchapterhead
%   \headtitle
%   \subheadtitle
%   \subsubheadtitle
%   \prepoint
%   \prefil
%   \postfil
%Box vars   (declared in blue.tex)
%   \prefacebox
%   \contentsbox
%New dimens (declared in blue.tex)
%   \generalindent
%   \gutter
%   \xshift
%   \yshift
%   \subsubheadindent
%New counts (declared in blue.tex)
%   \headcnt
%   \chaptercnt
%   \examplecnt
%   \exercisecnt
%New writes (declared in blue.tex)
%   \toc
%   \toe
%Fonts
%   \chapterfont\Large.........4
%   \subtitlefont..............5
%Chapter title matters
%   \chapno              (manmac)
%   \prechapterhead.......102-109
%   \postchapterhead......111-138
%   \beginchapterhead.....140-141
%   \storechaptertitle....143-144
%   \endchapterhead.......146-148
%   \chapterhead..........150-154
%   \setupappendices......156-159
%   \pasteupanswers.......161-165
%   \pasteuptoe...........167-176
%   \pasteuptoc...........178-191
%   \pagenorepresentation.193-194
%Summary
%   \beginsummary      (blue.tex)
%   \endsummary        (blue.tex)
%   \summary           (blue.tex)
%   \beginquote...........252-254
%   \endquote          (blue.tex)
%Header matters
%Head
%   \beginhead............302-304
%   \storeheadtitle.......306-307
%   \endhead..............309-310
%   \head.................312-315
%   \posthead.............317-324
```

[14] A detail in here is to provide for a toggle whether a number or a character should show up in the list.

[15] When no index is wanted the circumflex can be assigned category code 7 at the beginning of the script and all is fine.

```
%Subhead
%   \beginsubhead.........351-353
%   \storesubheadtitle....355-356
%   \endsubhead...........358-359
%   \subhead..............361-364
%   \postsubhead..........366-371
%Subsubhead
%   \beginsubsubhead......401-403
%   \storesubsubheadtitle.405-407
%   \endsubsubhead........409-410
%   \subsubhead...........412-417
%   \postsubsubhead.......419-424
%Inner markup
%   \begindemo         (blue.tex)
%   <escapechar>yields  (blue.tex)
%   \enddemo           (blue.tex)
%   \beginexample.........526-531
%   \storeexampletitle....533-534
%   \endexample...........536-544
%   \example..............546-549
%   \hang.....................553
%   \itemitem.............555-557
%   \textindent...........559-561
%   \exercise.............576-581
%   \answer...............583-587
%   \ansno................589-592
%Generalized \samplebox
%   \point              (blue.tex)
%   \gbox               (blue.tex)
%Centering verbatims
%   \boxlines           (blue.tex)
%   \begincenterverbatim(blue.tex)
%   \endcenterverbatim  (blue.tex)
%Preliminary pages matter
%   \def\beginscript......703-772
%Closing pages matter
%   \endscript............901-903
%   \pwtbackcover.........905-929
%Auxiliaries
%   \beginkeywords........802-803
%   \preacknowledgements..805-806
%   \pasteupacknowledgements808/812
%   \beginpreface.........814-815
%   \endpreface...............817
%   \preface..............819-822
%   \pasteuppreface.......824-825
%   \beginforewordtranslator827/9
%   \endforewordtranslator....831
%   \forewordtranslator...833-836
%   \pasteupforewordtranslator838/9
%   \begincontents........841-847
%   \endcontents..............849
%   \pasteupcontents..........851
%   NTG info          (tools.dat)
%Defaults..............1000-10xx
%   (\she.....................1037
%    \sshe...................1038)
%end Table of Contents
```

9 Acknowledgements

Many a suggestion with respect to simple markup tags in the user interface was done by Erik Frambach. Jos Winnink proofed the article, as usual. Thank you!

10 Conclusion

BLUe's `\report` format emerged because I needed it to format the user's guide 'Publishing with TEX.'

An important observation is that titles are reused a lot, and have to be stored. For storing I used the mechanisma of

- a box and paste up later
- toks variables, with the right category codes, and combined with
- writing to a file and including the file later.

The storing via an `\insert` similar to plain's footnotes is perhaps the way to allow for storing and processing on the fly, although the `\insert` mechanism has been designed with the OTR processing in mind. This has been postponed because I don't need nor wish verbatims in titles. In footnotes, yes I do, and that is very convenient. For marginnotes it might be relevant too. More on this later in 'Paradigm: Eating and preserving.'

Writing to a file and attaching page numbers to the entries needs awareness of the asynchronism of the OTR, in actual fact that a `\write` must be used and not an `\immediate \write`.

In one week I could write and format the articles 'BLUe's format Databases,' 'BLUe's Letters,' and 'BLUe's Reports,' next to polishing a little on the formats in BLUe's format system. The last article was correctly formatted immediately.[16]

At last, I reached the stage which I had mastered before with other programming languages: a blue script runs correctly when proofed for the first time.

After all those years I feel fairly confident with TEX.

References

The TEXbook and LATEX user's guide are omni-present and not explicitly listed. For my works consult lit.dat, via the use of `\search` for example.

[16] The bad news is that polishing the contents of the articles did cost me another couple of weeks. That has nothing to do with TEX, but all with the writing process as such, if not with the learning on the fly.

Paradigms: Two-part macros

Kees van der Laan

Hunzeweg 57,
9893 PB Garnwerd, The Netherlands
cgl@rc.service.rug.nl

1 BLUe's Design III

Hi folks. When attending Amy's class in 1990, I was much surprised about the two-part macro. In Algol, FORTRAN, PASCAL and ADA, I had not heard of the concept, let alone I was familiar with it.

In blue.tex they are at the heart of the syntax for the markup language. To speak with Jackowski 'I use them all the time.'

2 One-part vs. two-parts

One-part macros are explained in chapter 20 of *The TEXbook*. They are used as a shortcut for the replacement text parameterized by at most nine arguments.

A two-part macro is different. The first part sets up the 'environment' followed by script elements and ended by the second part, to finish up the environment. LaTeX emphasizes the environment concept in for example

```
\begin{abstract}...\end{abstract}
\begin{center}...\end{center}
\begin{itemize}...\end{itemize}
\begin{picture}...\end{picture}
\begin{quote}...\end{quote}
\begin{tabular}...\end{tabular}
\begin{thebibliography}...
          \end{thebibliography}
\begin{verbatim}...\end{verbatim}
%etc.
```

3 Why?

The need for bothering about two-part macros is that the enclosed script elements are processed on the fly, meaning with the right catcodes.

To digress a little on the above the following hypothetical example. Suppose we have

```
{\catcode`*=13
 \gdef\begindemo{\bgroup
   \catcode`*=13 \def*{MUL}}
\gdef\demo#1{\catcode`*=13 \def*{MUL}#1}
}\let\enddemo\egroup
```

then the result of

```
\begindemo*\enddemo
%and
\demo*
```

is different. The first yields MUL and the latter *.

Explanation

In the two-part case the * is seen after the catcode has changed. while in the latter the * is seen, and the catcode *fixed*, before it is made active.

However, in chapter 20 of *The TEXbook* there is no treatment of two-part macros, nor is there an entry for it in the index, alas. Exercise 5.7 deals with named blocks and checking of them. The latter is used in LaTeX to make sure that the right environment closing tag is used in the markup. In Appendix E, where example formats (o.a. manmac) are explained, two-part macros are abundant, for example

```
\beginchapter...\endchapter
\beginlines...\endlines
\begindisplay...\enddisplay
\begintt...\endtt
\begimathdemo...\endmathdemo
\beginchart...\endchart
\beginsyntax...\endsyntax
\begindoublecolumns...\enddoublecolumns
\exercise...\answer...\par
```

Furthermore, of late two questions were posed on TeX-nl, which exposed the unfamiliarity with two-part macros. All this was enough for me to spend a paradigm column on two-part macros.[1]

Example (`\beginlines...\endlines`)

The functionality is that the script in between is processed line-by-line and preceded and followed by an `\hrule`.

```
\def\beginlines{\par\begingroup\nobreak
   \medskip\parindent0pt\hrule\kern1pt
   \nobreak\obeylines\everypar{\strut}}
\def\endlines{\kern1pt\hrule\endgroup
   \medbreak\noindent}
```

In the TEXbook script this is combined with in-line verbatim.[2]

Explanation

The replacement text of `\beginlines` is processed, followed by the formatting on-the-fly of the inserted material (after `\beginlines`) up to `\endlines`. The replacement text of the latter finishes it up.

[1] Note that `\beginchapter`'s title is not processed on the fly. In the 'Paradigm: Headache?' I have shown how the title and the contents of the chapter can be processed on the fly.

[2] To set text verbatim. By the way, this is another approach to 'verbatims with an escape character.'

Unwanted breaks are avoided. The `\hrule` is set in the first part and in the second part next to opening and closing of the group. The in between script is processed with `\obeylines` on.

Example *(*`\begindisplay`. . . `\enddisplay`*)*

The functionality is that the script in between is processed as a non-centered display, indented by `\displayindent`, next to the value of `\parindent` from the template. Pruned from non-essential issues for the two-part macro idea, the macros read as follows.

```
\def\begindisplay{$$\the\thisdisplay\halign
   \bgroup\indent##\hfil&&\qquad##\hfil\cr}
\def\enddisplay{\crcr\egroup$$}
```

Explanation

The replacement text of `\begindisplay` is processed, followed by the formatting on-the-fly of the inserted material (after `\begindisplay`) up to `\enddisplay`. The replacement text of the latter finishes it up.

$$ followed by `\halign` is something special. It starts the so-called alignment display, meaning that each hbox of the `\halign` is added to the main vertical list indented at the left by `\displayindent`. It is *not a math display*. `\the\thisdisplay` allows to insert assignments.

By the way, note that the user is not bothered by the details of the template of the `\halign`; it is already there.[3]

And what about a one-part on top?

This is not possible via my method as explained in 'Paradigms: Headache?', because each table entry must have balanced braces. Suppose we have

```
\def\display#{\begindisplay\bgroup
   \aftergroup\enddisplay
   \let\dummy=}
```

then the `\bgroup` after `\begindisplay` is 'unbalanced' in the first column, except when it is about one entry only.

```
\display{a}  %works
\display{a&b}%doesn't work
```

I let it go

because I could not provide a nice solution. What I tried is out of balance with just using the two-part macros. The best I could get at, when we allow in-line verbatim, needs the following input.

```
\thisdisplay{\catcode`\!=0 \catcode`\\=12 }
\display{\a&b!cr e&f}
```

Conclusion

When tables are involved my method of building one-part macros on top of two-part macros is not suited.[4]

For the manmac

version of the two-part macro see *The TEXbook* 421. Note that there the `\catcode`\^^M` annihilates the effect of `\obeylines`. The `\obeylines` was introduced only to allow for an optional argument. Because of my `\thisdisplay` toks variable, the `\obeylines` and its annihilator are no longer needed. Knuth's coding has been simplified, at the expense of introducing a token variable `\thisdisplay`.[5]

4 From the TeX-nl list

Andrea de Leeuw van Weenen and Ton Biegstraaten posed the following problems.

- let characters print other characters
- let _ in math denote an underscore and not a subscript.

Although it turned out that my suggestions are not the 100% required ones, I'll expose them here nonetheless, because they illustrate the use of two-part macros.

Andrea's problem

Let us suppose that the problem is to let B typeset 1, on demand. Then a solution reads.

```
\def\beginIT{\bgroup\catcode`\B=13 \ITstart}
{\catcode`\B=13
 \gdef\ITstart{\def B{\char'61}}}
\def\endIT{\egroup}
%with use
ABC\quad
\beginIT ABC\endIT\quad
ABC
```

The result reads ABC A1C ABC.

The problem which remained is that Andrea needs simultaneously macros with those letters like B in their name. She added the problem to her list of 'Impossible with TEX problems.[6]'

Ton's problem

The restriction, which made that my solution was not appropriate, is that it should be possible to use the solution as argument of one-part macros, and that to unlimited depth.[7] In my approach all involved one-part macros had to be rewritten into two-part ones. However, if people would start to think in two-part macros (nearly) all would have been fine.

```
\def\beginusn{\hbox\bgroup\catcode`\_=13
   \startusn}
```

[3] In my `\btable` macro, I allowed the possibility for a user to supply his own template, because I stored the template in a token variable.

[4] If one prefers a simple, but *restricted* one-part macro provide `\def \display`#1{ `\begindisplay` #1 `\enddisplay`}.

[5] Optional arguments — well, more generally 'Parameterization' — will be subject of the next paradigm column.

[6] I'm curious to see that list in MAPS some day.

[7] Courtesy Piet van Oostrum.

```
{\catcode‘\_=13\gdef\startusn{\def_{\_}}}
\def\endusn{\egroup}
%with use
$a_b\quad \beginusn a_b\endusn\quad a_b$
```

and result

a_b a_b a_b.

On top of the above two-part macros we can add one-part macros *wih the same functionality*, as explained in the ‘Paradigm: Headache?’

The one-part macros read `\def\IT{\beginIT\bgroup`

```
   \aftergroup\endIT
   \let\dummy=}
%
\def\usn{\beginusn\bgroup
   \aftergroup\endusn
   \let\dummy=}
```

As expected `ABC\quad\IT{ABC}\quad ABC`
yields ABC A1C ABC, and
`$a_b\quad\usn{a_b}\quad a_b$`
yields a_b a_b a_b.

Note that I omitted here the # as last element of the parameter list, neglecting some built-in security checks.[8]

5 `\eqalign` as two-part macro

As an example of how to cast a one-part macro into two parts, and a one-part macro[9] on top, let us rewrite `\eqalign`, *The TEXbook* 362. The extra functionality of this approach is that the two-part variant can be used in those cases where the argument needs to be processed on the fly.

```
\def\begineqalign{\,\vcenter\bgroup
   \the\thiseqalign\openup1\jot\m@th
   \starteqalign}
\def\starteqalign{\ialign\bgroup
   \strut\hfil$\displaystyle{##}$&&
   $\displaystyle{{}##}$\hfil\crcr}
\def\endeqalign{\crcr\egroup\egroup}
%with the one-part
\def\eqalign#1{\begineqalign
   #1\endeqalign}
```

I don’t have a concrete example for the need for modifying `\eqalign` towards processing on the fly. However, it illustrates how to rewrite a one-part macro into two-parts as basis.

Looking back

I like the consistent markup via

`\begin`⟨*tag*⟩ ⟨*copy proper*⟩ or `\`⟨*tag*⟩`{`⟨*copy proper*⟩`}`
`\end`⟨*tag*⟩

The right-hand variant is suited for the markup of headings, for example. It has been adopted in blue.tex, as basic syntax, for the markup language.

6 Multiple use of copy

Sometimes we need to process the copy — or should we talk about data then? — more than once. An example is the data for a crossword, where I used the data for typesetting the puzzle — the data reflect the structure — and the solution. See ‘Typesetting crosswords via TEX, revisited,’ MAPS 92.2.

The basic idea is to store the data with the right catcodes.[10]

```
\def\bdata{\begingroup
   \obeylines\obeyspaces\store}
\def\store#1\edata{\endgroup
   \def\storeddata{#1}}
```

Explanation

The data, in natural markup line-by-line, can be supplied between `\bdata` and `\edata`. The `\edata` is a parameter separator and not the invocation of the closing part of a two-part macro, although it looks the same. What happens is that `\bdata` sets up the environment, especially provides the right catcodes. `\store` ends the environment (scope) and stores the data, with the wanted catcodes, as replacement text of `\storeddata`. In order to appreciate the subtleness of the above coding the following digressions.

6.1 Two-part macros and storing on the fly

This is inhibited by the following[11]

- the *opening and closing brace* of the replacement text of a `\def` must be explicit
- the right-hand side of a token list assignment must be explicit.

The following innocent coding is therefore incorrect.[12]

```
\def\bdata{\begingroup
   \obeylines\obeyspaces
   \gdef\storeddata\bgroup}
\def\edata{\egroup\endgroup}
```

Possible alternatives to my coding above are

```
\def\data{\obeylines\obeyspaces
  \gdef\storeddata}
%with use
\begingroup
\data{ab c
```

[8] In the case of the # end separator the text after the macro invocation must syntactically begin with an opening brace. When the # separator is omitted, anything can follow ⟨tag⟩.

[9] Not more limited than the one available.

[10] Perhaps the most trivial approach is to insert the data each time we need it. I consider that inelegant and also error-prone. The given macro is a beautiful example, if I may say so, of what Victor Eijkhout and David Salomon call two-step macros (see later), while at the user level the macro can be used as if it is a two-part macro, with the nice opening and closing tags.

[11] Courtesy Victor Eijkhout in ‘TEX by Topic,’ section 10.3 group delimiters. Awareness of these restrictions is indispensable for writing two-part macros. I omitted the use of `\setbox`, because once set in a box one can’t do much with the data anymore.

[12] The use of explicit braces is incorrect as well.

```
      e fg}
\endgroup
%
%and via the use of a toks variable
%
\newtoks\storeddata
\def\data{\obeylines\obeyspaces
  \global\storeddata}
%with use
\begingroup
\data{ab c
      e fg}
\endgroup
```

Nice aspects of the above approaches are

- at the outer level I abstracted from storing in a def or a token variable, and
- the symmetry.

Definitely not nice aspects are

- it looks as if the data are stored in `\data`, and
- the `\begingroup` and `\endgroup` at the user level.

6.2 A one-part on top?

My scheme does not work for this case. Some puzzling yielded as one-part `\data` on top of `\bdata`, with `\edata` eliminated.[13]

```
\def\data{\begingroup
    \def\store##1{\endgroup
       \gdef\storeddata{##1}\endgroup}
    \bdata}
%With use
\data{ab c
      d ef}
```

Explanation

`\data` starts a group and (re)defines `\store`. The invocation of `\bdata` set the catcodes — via `\obeylines` and `\obeyspaces` — and invokes `\store`. The argument to the latter macro is stored in `\storeddata` with the right catcodes. `\store` also ends the groups.[14]

6.3 Chapterhead

For blue.tex I designed `\report`. A report takes chapter titles. The problem is: How to write macros consistent with the philosophy of starting from two-part macros and building a one-part on top, *with* the chapter title also stored for use in the running headline, for example.

In an abstract sense this is equivalent to the `\bdata` `\edata`, `\data` suite. It is even simpler, because I just have to store the name and allow the following use.

```
\beginchapterhead
<name>            or    \chapterhead{<name>}
\endchapterhead
```

The required result must be such that the chapter name will be typeset appropriately within context, as prescribed by the token variables `\prechapterhead` and `\postchapterhead`, and that the name will be stored in the token variable `\chaptername`.

The coding of the two-part macros read.

```
\def\beginchapterhead{\the\prechapterhead
   \storechaptername}
\def\storechaptername#1\endchapterhead{%
   \chaptername={#1}\endchapterhead}
\def\endchapterhead{{\chpfont
   \the\chaptername}\the\postchapterhead}
```

The one-part macro on top reads.

```
\def\chapterhead{\bgroup
   \def\storechaptername##1{\egroup
     \global\chaptername={##1}%
     \endchapterhead}
   \beginchapterhead}
```

The head-suite of macros also need processing and storing if not for writing to a ToC file. The use of the token variable `\prechapterhead` provided the hook to change the catcode of the circumflex — which in blue.tex is default active because of preparing Index Reminders — into 7 and allow processing math as part of the title.

What have we gained?

We can use now the title with different fonts, as title and in the running head. Moreover, we can use the `\beginchapterhead`, `\endchapterhead` pair to enclose the title, or let it look as an assignation to `\chapterhead`. Looking back there emerged a paradigm for the use

```
\begin<tag>
<copy>            or    \<tag>{<copy>}
\end<tag>
```

with `<copy>` also stored in the token variable `\tagname`. Useful!

6.4 And what about multiple use with different catcodes?

Like Knuth we are at loss, unless we make use of a file. It occurs in manmac's math demos, for example

Input	*Output*
`$x^2$`	x^2

needs markup with repetition of the data[15]

```
\beginmathdemo
   \it Input&\it Output\cr
   \noalign{\vskip2pt}
   |$x^2$|%<---
   &x^2   %<---
\endmathdemo
```

Subtle, very subtle. One thing is crystal clear, however. Because of the above varieties (and pitfalls?), a discipline of TEX coding is needed.

[13] Note that in-line verbatim as part of the data goes wrong, in the sense of unexpected results.

[14] The group opening in `\bdata` is not needed here, but within the context of the two-part macro next to the one part, it is needed.

[15] Borrowed from *The TEXbook* script. In blue.tex I added `\crcr` to `\endmathdemo`, for consistency with `\halign` use.

7 Epilog

Eijkhout in 'TEX by Topic' and Salomon in 'Insights and Hindsights' treat *two-step* macros, not *two-part* macros.[16] One macro will set up conditions and a second will do the work. The difference with two-part macros is that the 'workmacro' also terminates the conditions, while in two-part macros the second part has only the functionality to terminate. Probably other macros are involved to do the work. A beautiful example from manmac is the non-centered display macro with tags

```
\begindisplay %to set up conditions
\startdisplay %to do the work
\enddisplay   %to finish up
```

As known, I prefer — like Knuth — the separation of concerns principle, and like opening and closing tags.

To my knowledge it is not possible to build gracefully, and with the *same functionality*, a one-part *table* macro on top of its constituent two-parts, in full generality.

Have fun, and all the best.

[16] Apparently they did not inspect manmac in detail. In Eijkhout's book look at section 11.9.4, the macro `\PickToEol`. In Salomon's courseware look at section 5.19, the macro `\elp`.

Paradigms: Parameterization I

Options

Kees van der Laan

Hunzeweg 57,
9893 PB Garnwerd, The Netherlands
cgl@rc.service.rug.nl

1 BLUe's Design IV

Hi folks. Much parameterization in TEX has already been taken care of via the various modes of TEX, that is by the states script elements can be processed in. How ingenious and handy that may be, it is not sufficient.
Knuth also provided switching commands for fonts, formats and language.[1]
Then there are the various parameters—integer, dimen, glue, muglue—as listed in *The TEXbook* 272–274.

Markup tags can be parameterized via (at most 9) arguments. However, there exist also for this purpose (global) token variables—the `\every<tag>`s—with `\output` and `\errhelp` as special cases.

To remind the reader of Knuth's `\every`s, I have enumerated them below.

```
\everypar
\everymath, and \everydisplay
\everyhbox, and \everyvbox
\everyjob
\everycr
```

For using them, it is important to know where the token variables are precisely inserted.[2]

For my taste Knuth's `\every<tag>`s have not been appreciated as they should have been, the more so when extended by the analogons `\this<tag>`s. This in contrast with the abundant use of arguments, especially the so-called optional arguments.

1.1 Why?

The purpose of this note is to show that optional arguments have led to cumbersome TEX coding, while the same functionality can be attained more easily via the use of `\this<tag>`s. In general one can devote a monograph to markup language parameterization, I guess.

2 Examples of use of `\every`s

I looked through the *The TEXbook* for examples of use by Knuth himself.

2.1 `\everypar`

An example of use is given in Appendix D 381, about verbatim listings with line numbers. In blue.tex I enriched this approach by allowing selective line numbering. I mean by the latter that I can number parts, starting with any suitable number. This is handy when macro sets are accompanied by a table of contents (as should always be the case), which contains references to the line numbers of the macro set. From blue.tex the following.

```
\def\setupverbatim{\makeactive\‘%
  \let\!=!\makeescape\!%Knuth&Levy
  \def\par{\leavevmode\endgraf}%381
  \obeylines \uncatcodespecials
  \obeyspaces}
%
\def\numvrb{\vrblin0
   \everypar{\advance\vrblin1
    \llap{\sevenrm\the\vrblin\quad}}}
%
\def\nonum{\everypar={}}
!endverbatim
%
```

Note that the line numbers can be adjusted via modifying the counter `\vrblin`.

Another example is provided on 393, in Paragraph maneuvers. It is about automatically inserting `\hangindent` and `\hangafter`.

In manmac it is used in

- `\endchapter`, as `\everypar{\sl}`, to set the quotations slanted.
- `\beginlines`, as `\everypar{\strut}`, to insert a strut.

2.2 `\everydisplay`

This is used in the solution of exercise 19.4 to obtain non-centered displays.

```
\def\leftdisplay#1$${\leftline
 {\indent$\displaystyle{#1}$}$$}
\everydisplay{\leftdisplay}
```

This is refined in Appendix D 376, to allow for equation numbers as well. In 'Math into BLUes' I have worked on the Appendix D version, and also indicated how a single formula can be non-centered. This is also included in blue.tex.

[1] For example `\eightpoint`, `\report` (well in blue.tex), and `\language1`. Similar to the font switching macros I used `\english`, `\dutch` and the like, to activate all language dependent parameters for typesetting bridge with the right words. Early LATEX had several words 'hardwired' in the code in English. The latter was the reason for Johannes Braams to give birth to Babel, to parameterize and concentrate language specific issues, and to allow easy switching from one language to another.

[2] For example, in a setbox the right-hand side of `\afterassignment` is inserted after the opening brace of the box and followed by the tokens of `\everyhbox` or `\everyvbox`.

2.3 \everyjob

The example is exercise 24.5. In Salomon's courseware an example is given to open automatically files at the beginning of each TEX job.

2.4 \everycr

A practical example is given in *The TEXbook* 140, to inhibit a page break.

```
$$\everycr{\noalign{\penalty10000}}
\halign{\indent#\hfil\qquad&...\cr
If a letter is in style&then...\cr
\noalign{\vskip 2pt}
$D,D',T,T'$&text size&\cr
$S,S'$&script size&\sevenrm\cr
$\SS,\SS'$&scriptscript size&
                \fiverm\cr}$$
```

It is also used in plain, Appendix B 362, in the macro \displ@y.

in *The TEXbook* file I did not find examples of use of \everymath, \everyhbox, and \everyvbox.

In blue.tex I introduced \everyscript. My use is to allow for more than one script to be processed, via \everyscript{\notlastscript}. I also introduced \everyverbatim, and some more.

3 Options via \this< *tag* >s

First an example. I used it for the first time with verbatims.

```
\thisverbatim{\emc}%enable metacode
\beginverbatim
\def\⟨tag⟩{...
   ...}
!endverbatim%! is the escape character
```

Not only can metalinguistic variables be handled nicely within verbatims, but also the changing of catcodes and file verbatim inclusion go easy with the use of these token variables.

The verbatim suite of blue.tex can be used with (LA)TEX, because the mechanism is simple, and not in conflict. File verbatim inclusion goes as follows[3]

```
\thisverbatim{\input ⟨file⟩}
\beginverbatim
Some text after the file.
!endverbatim
```

4 The parsing of options

Eijkhout in 'TEX by Topic' gives a template for coping with optional parameters.[4] The work is done by \OptArgCom, with as optional argument either the default or the one supplied. The markup starts with \Com followed either by a left bracket—a convention to start an option—or the argument.

```
\def\Com{\futurelet\testchar\MaybeOptArgCom}
\def\MaybeOptArgCom{\ifx[\testchar
   \expandafter\OptArgCom\else
   \expandafter\NoOptArgCom\fi}
\def\OptArgCom[#1]#2{...}
\def\NoOptArgCom{\OptArgCom[⟨default⟩]}
%with use
\Com[...]{...}    or    \Com{...}
```

With the concept of \thisCom the coding template and markup might look like the following.

```
\thisCom{⟨default⟩}
\def\Com#1{...\the\thisCom...
   \thisCom{⟨default⟩}}%restore default
%with use
\thisCom{...}%Provides option(s), if any
\Com{...}
```

Explanation

The contents of the token variable is available to the macro. It can be virtually anything. At the end of the macro the default is restored. IMHO, with all respect, the latter approach is simpler than the parsing of optional arguments.

5 Head example

blue.tex allows as markup

```
\beginhead...\endhead     or    \head{...}
```

And what about the coding? In the 'Paradigms: Headache?' I have shown the blue.tex coding

```
\def\beginhead{\the\prehead\bgroup\headfont}
\def\endhead{\egroup\the\posthead}
%with auxiliaries
\prehead{\vskip0pt plus2ex
   \penalty-250\vskip0pt plus1ex
   \bigskip\noindent}
\posthead{\medskip\nobreak
   \noindent\ignorewhitespace}
%and minimal variant
\def\head#{\beginhead\bgroup
   \aftergroup\endhead
   \afterassignment\ignorespaces
   \let\dummy=}
```

Note that in blue.tex, I also introduced the token variables \pre<tag>, and \post<tag>, to parameterize the placement within context. The advantage of doing so is that these tags have only one function and can therefore be customized easily.[5]

5.1 Relation with tugboat.sty

Submissions for TUGboat need as markup for headings

```
\head...\endhead     or    \head*...*
```

where in the short variant the spaces around the *s are neglected. TUGboat provides as toplevel coding

```
\def\head{\begingroup
   \def\CurrentTag{head}%
   \@allowindentfalse
   \@defaultoptions
   \@savingargumenttrue
   \def\\{\break}%
   \@checkoptions}
```

[3] It is true, that I need one escape character.

[4] I adapted the macros a little.

[5] A white lie. In \report format I also reused the headtitle in the ToC, ToE, and in the running header.

```
\def\endhead{\endgraf
   \ifcase\headlevel\or
      \@domainhead  \or
      \@dosubhead   \or
      \@dosubsubhead\fi
   \endgroup\@next}
```

What can be seen from the above is that both codings are based on two-part macros. The coding for TUGboat uses general mechanisms not restricted to `\head`, and is therefore difficult to understand.[6] Especially, when one realizes that next to the parsing of options, the minimal variant is also handled. (The parsing looks for *s). Clever, very clever. But the functionality can be attained simpler, with the extra bonus of easy maintenance and customization, IMHO, with all respect.

5.2 Relation with ams.ppt

Submissions to AMS (in ams.ppt) need as markup for headings

```
\head...\endhead
```

The coding reads

```
\outer\def\head#1\endhead{%
   \add@missing\endroster
   ...
   \add@missing\endproclaim
   \penaltyandskip@{-200}\aboveheadskip
   {\headfont@\raggedcenter@
    \interlinepenalty\@M
    #1\endgraf}\headmark{#1}%
   \nobreak\vskip\belowheadskip}
%
\let\headmark\eat@
```

The `\end@missing` checks whether the head occurs within the environment. `\eat@` gobbles its argument. There is no `\nofrillscheck`. This is done, however, in `\subhead`. The coding of the check is difficult to understand, not in the least because it allows for options.[7]

5.3 LaTeX's `\section`

The style defines `\section` as an invocation to `\@startsection`. From latex.doc the following.

```
%\@startsection{name}{level}{indent}
%  {beforeskip}{afterskip}{style}
%  optional * [altheading]%
%  {heading}
%Generic command to start a section.
%Name  : e.g. subsection
%Level : a number, denoting depth of
%        section e.g., chapter=1,
%        section=2 etc.
%Indent: Indentation of heading from
%        left margin
%Beforeskip: Absulute value is skip to
%           leave above the heading.
%           If negative, then paragraph
%           indent of text following
%           heading is suppressed.
%Afterskip : if positive then skip to leave
%           below heading, else negative of
%           skip to right
%           of run-in heading.
%Style     : commands to set style.
%If * missing then increments the counter. If
%it is present, then there should be no
%[altheading] argument. Use the counter
%'secnumdepth' whose value is the highest
%section level that is to be numbered.
```

This is just the top of the ice-mountain, but generic it is for sure. My choice would be the following. To go first for making it as simple as possible, and perhaps if the need is still there, use generic coding in order to save on development and maintenance costs.

6 Knuth's options after begin tag

There is a beautiful example of allowing for options in manmac. It is used in *The TeXbook* 29. All what follows after the opening tag `\begindisplay` up to the end-of-line is taken as (optional) argument, and inserted in the alignment display between $$ and `\halign`.

```
\begindisplay\hbadness10000
\hbox spread-.666667em{The badness
   of this line is 100.}&
     \quad(very tight)\cr
...
\enddisplay
```

This example shows what Knuth had on his mind with respect to options and explains why he did not introduce `\this<tag>`s. The coding is a litlle more complicated but systematic.

Can it be applied applied throughout a format? With `\beginverbatim` I stumbled upon problems in applying options to the in-line verbatim `|...|`.

7 Epilog

The paradigm is that too general, monolithic, codes are difficult to understand and to maintain. Borrowing macros from these collections is near to impossible. `\this<tag>` is a simple alternative for coping with optional arguments, and is used in blue.tex.

Knuth suggested a solid naming convention for two-part macros: `\begin<tag>`, `\end<tag>`, and for the second step macro `\start<tag>`, if any.

This naming convention has been adopted in blue.tex, next to `\<tag>` for one-part minimal variants.

Have fun, and all the best.

[6] An example is finding the answer to the question whether the 'argument' is processed on the fly. `\@savingstrue` suggests that it is stored. Try to confirm the assumption in the code, and you will agree that even reading the code is difficult, let alone to adapt it. To borrow macros for use in other collections is also inhibited, as Wietse Dol communicated to me. Of course, it is a good thing to go for general mechanisms in macro writing. But the right balance between metacodes and single shot coding is the royal road, IMHO, with all respect.

[7] I was told that AMS considers to abandon `\nofrills` altogether.

BLUe's Typesetting of Pascal

Lean and Mean

Kees van der Laan

Hunzeweg 57,
9893 PB Garnwerd, The Netherlands
cgl@rc.service.rug.nl

Abstract

The formatting of (partial) PASCAL programs within plain TEX is proposed. Only `\beginpascal` and `\endpascal` have to be added as markup. In general the literate programming tools are to be preferred, especially when designing, developing, documenting, and maintaining professional software.

Keywords: Compatible extension, education, fifo, macro writing, Pascal texts, pattern matching, plain TEX, pretty-printing, reusable software parts, software engineering.

1 Introduction

Programs are meant for computers and humans to be understood. Therefore publishing a program is not just a verbatim listing. In general the typesetting of a program is part of the documentation process.

Knuth with his (CWEB) literate programming tools aimed at programming as a craft, to program

- with less errors, and
- to create documentation simultaneously.

Pretty-printing of code as such is an academic exercise. Spending too much energy on it is suboptimization, because of the wealth of already available tools.

1.1 Why?

Knuth, *The TEXbook* 234 and ex18.9, illustrates the problem via a small program fragment, especially in view of non-monospaced types. He adds tabular markup to secure vertical alignment, and inserts tags for marking up for fonts. If only program fragments are the subject, this way of doing is OK. The larger the code the more tedious hand markup becomes. Moreover, detailed hand markup is inconsistency-prone.

I strove after to leave the program as such unaltered, and assumed that the Alignment has already been taken care of. The fonts for the reserved words, the program text, and can be adaptedly easily by the user.

1.2 Disclaimer

Because I refrained from inserting markup in the program, vertical alignment can be hampered, due to the use of the non-monospaced fonts and different representations of some symbols like $\geq$.

[1] Courtesy Wilson & Addyman.

1.3 Notations and definitions

`\ea` denotes `\expandafter`. `\nx` denotes `\noexpand`. FIFO stands for First In First Out, as described in my 'FIFO and LIFO sing the BLUes.'

2 Use

Within the context of blue.tex the following example[1]

{ Print length of third side of triangle

given two sides and enclosed angle }

program *EX4A(input, output)*;

var *a, b, c, angle* : *real*;

begin *read(a, b, angle)*;

c := *sqrt(sqr(a) + sqr(b) − 2 ∗ a ∗ b ∗ cos(angle))*;

writeln('The third side is', c)

end.

results from

```
\beginpascal
{Print length of third side of triangle
 given two sides and enclosed angle}
program EX4A(input, output);
var a, b, c angle : real;
begin read(a, b, angle);
   c:=sqrt(sqr(a) + sqr(b) - 2*a*b*cos(angle));
   writeln('The third side is', c)
end.
\endpascal
```

With the token variables `\prepascal` and `\postpascal` the placement within context can be altered. For example to 'center' the program use the following.

```
\prepascal{$$\vbox\bgroup\hsize.5\hsize}
```

```
\postpascal{\egroup$$}
\beginpascal
⟨code⟩
\endpascal
```

3 What are the problems?

Given that the layout in ASCII is already there, then the vertical alignment is violated with non-monospaced fonts.[2]

If we adopt Knuth's approach to add markup for alignment then the program is altered, and it doesn't compile as such.[3]

Doumont proposed to use TEX as a pre-processor, a full-blown pretty-printer. Tradition has it that the layout is done by other tools than TEX. Doumont's approach is therefore not simple, and not easily adaptable to other languages. My modest approach is hopefully a good balance between Knuth's markup of program fragments and Doumont's pretty-printer.

4 Coding

My approach is to process a program line-by-line, and each line word-by-word, an application of nested FIFO processing.

Comments and special symbols can do their job by making them active and defining them appropriately.

5 Customization

The user can choose his fonts. Default is provided

```
\let\reservedwordsfont\bf
\let\pascalprogramfont\it
\let\pascalcommentfont\rm
```

The setup can be used for similar languages by adapting `\reservedwords` and the macros for typesetting the special symbols.

6 Availability

The macros — ≈ 75 lines — are shareware and come with blue.tex, i.e., are included in the tools.dat database.[4]

7 Acknowledgements

Erik Frambach thank you for your assistance, and your sound judgement. As usual Jos Winnink helped me to procrust the markup of the article into `maps.sty`. Jean-Luc Doumont prompted the subtitle.

8 Conclusion

In general it is difficult to know when to stop, as Schumacher in his precious 'Small is beautiful' already pointed out. Perhaps, I stopped too early, but the macros do what I want them to do. My case rest.

9 References

I learned from Doumont, but my approach is much different. Look in lit.dat for more details of consulted work.

10 Appendix: The macros

To facilitate the use blue.tex contains the storage allocations and the following.

```
\def\beginpascal{\begingroup\let\pascaltool=x
   \loadtool\afterassignment\fifol\let\dummy=}
```

As part of tools.dat are provided the blue collar macros. Note that I included in the set of reserved words `\end.` and `\end;`.

In the following driver I have collected the items from blue.tex and tools.dat to provide a complete and independent set to be used with AnyTEX.

```
%Declarations
\newtoks\prepascal
\newtoks\postpascal
\newtoks\reservedset
\newcount\commentstatus
\newif\iffound
%Initializations
\let\reservedwordsfont\bf
\let\pascalprogramfont\it
\let\pascalcommentfont\rm
\prepascal{\medskip\noindent}
\postpascal\prepascal
\let\ea\expandafter
%Auxiliary
\newif\iffound
\def\loc#1#2{\def\locate##1#1##2\end
 {\ifx\empty##2\empty\foundfalse
 \else\foundtrue\fi}\ea\locate#2.#1\end}
%
\def\beginpascal{\begingroup\parindent0pt
%   \let\pascaltool=x \loadtool
   \afterassignment\fifol\let\dummy=}
{\obeylines\gdef\fifol#1
 {\ifx\lofif#1\lofif\fi%
  \processl{#1}\fifol}}
\def\wofif#1\fifow{\fi}
%From tools.dat
\def\lofif#1\fifol{\fi
   \the\postpascal\endgroup}
\let\endpascal\lofif
\def\processw#1{%
 \ifnum\commentstatus>0 #1\else
  \ifx\relax#1\relax{ }%space
  \else\foundfalse
   \loc{#1}{\the\reservedset}%
    \iffound{\reservedwordsfont#1}%
    \else\if=#1${}={}$\else
     {\pascalprogramfont#1{}}\fi\fi
 \fi\fi}%end processw
%Typesetting : ;= ; . ^
\def\becomes{:=}
\catcode`\:=13
\def:#1{\if=#1${}\becomes{}$\else
            {\rm\char`\:}#1\fi}
\catcode`\;=13
```

[2] I don't care because readability is not hampered. The rule for indentation will not be obeyed.

[3] I know of Rein Smedinga's argument that for small educational fragments this is not a problem.

[4] blue.tex is on NTG's 1995 4AllTEX CD-ROM, and will be offered to the CTAN. Each happy user from blue.tex is requested to send me $25 to maintain my hardware. The donor will be added to my register.blu database.

```
     \def;{{\/\rm\char‘\;}}
\catcode‘\.=13 \def.{{\pascalprogramfont
                     \char‘\.}}
\catcode‘\^=13 \def^{\char‘136}
    %\def^{{\enspace$\uparrow$}}
%Reserved words
\reservedset{and array begin case const
 div do downto else end. end; file for
 function goto if in label mod nil not of
 or packed procedure program record repeat
 set then to type until var while with}%
%Handling of >= <= <> > <
\def\lt{<}\def\gt{>}
\catcode‘\>=13
\def>#1{\if=#1${}\geq{}$\else
             ${}\gt{}$#1\fi}
\catcode‘\<=13
\def<#1{\ifx>#1${}\ne{}$\else
       \if=#1${}\leq{}$\else
       ${}\lt{}$#1\fi\fi}
%Handling of + - *
\def\-{-}%Save for hyphen use
\def\plus{+}\def\minus{-}\def\times{*}
\catcode‘\+=13 \def+{${}\plus{}$}
\catcode‘\*=13 \def*{${}\times{}$}
%{\catcode‘\-=13 \gdef-{${}\minus{}$}}
%Later catcode is set for minus
%Last part of processing
\obeyspaces\let =\
\def\fifow#1 {\ifx
\wofif#1\wofif\fi\processw{#1}\ \fifow}%
\def\processl#1{\fifow#1 \wofif
 \endgraf}%First space is needed
%
%Keeping track of comments, and
%using comment fonts
\catcode‘\{=13\catcode‘\}=13%
\catcode‘\[=1\catcode‘\]=2%
\gdef{[$\{\,$\global%
\advance\commentstatus1\relax%
\pascalcommentfont]%
\gdef}[$\,\}$\global%
\advance\commentstatus-1]%
\catcode‘\-=13 \def-[$[]\minus[]$]%
\obeylines\smallskip%
\the\prepascal\relax%
```

TUG'95

Preliminary Program

St. Petersburg Beach, Florida

July 24 – 28

↪ **Monday a.m.** ↩

TEX and METAFONT techniques: We start with a good burst of traditional TEX papers, hearing about the remarkably different ways TEX and METAFONT are developing:

Richard J. Kinch: METAFOG: converting METAFONT shapes to contours;
Jiří Zlatuška: When METAFONT does it alone;
Gabriel Valiente Feruglio: Modern Catalan typographical conventions;
Jose Luis Ruiz: Complex labels, legends, and composition of figures in TEX documents;
Petr Sojka: Notes on compound word hyphenation in TEX;
Yannis Haralambous: ScholarTEX version 1.

↪ **Monday p.m.** ↩

TEX practical workshops: This is where people get up and actually *explain* how to use a package, and how you can get started with it. This will probably be organized as parallel sessions, possibly repeated later in the conference. Topics that are likely to be covered include:

PSTricks, the package to access almost all the power of PostScript from within TEX; with its developer, **Tim van Zandt**;
4AllTEX, the first (and only!) plug'n'play TEX system for DOS on a CD, with unrivaled features and fun; with one of its authors, **Wietse Dol**;
MusicTEX, high-quality music typesetting with TEX;
XyMTEX, sophisticated chemistry structures;
$\mathcal{AMS}$-LATEX, the best add-on for mathematical LATEX users;
PostScript fonts, how to set up new font families; with the developers of `fontinst` and PSNFSS;
MetaPost, the drawing package based on METAFONT which Knuth himself uses; introduced by its author, **John Hobby**.

Anyone who wants to offer a session here should contact Sebastian Rahtz (`s.rahtz@elsevier.co.uk`).

↪ **Tuesday a.m.** ↩

TEX Futures: The TEX world is not standing still: there are several projects which build on standard TEX to try and solve old and new problems. This session will give participants the opportunity to listen to progress reports from, and have detailed discussion with, members of the development teams.

Omega, a 16-bit TEX with sophisticated input/output filters, and the complementary Unicode font project, will be on show by its authors, **Yannis Haralambous** and **John Plaice**;
NTS, the New Typesetting System;
e-TEX, the first release of the extended TEX software; support programs like *MakeIndex*, *BibTeX* and *Patgen*, that are all at present being extended and developed.

↪ **Tuesday p.m.** ↩

Free for an exciting outing!

↪ **Wednesday and Thursday** ↩

Real World TEX, and electronic publishing: These two days will be devoted to traditional publishing concerns, how to get TEX working, and how it will be used in electronic publishing. Among the activities you can expect during these days are: What's new in electronic publishing? Is TEX playing a part? How can we preserve our hard work in the age of the Internet? Papers offered include:

T.V. Raman: An audio view of LATEX documents — Part II (Raman's thesis on this subject, presented recently won a national award);
Sebastian Rahtz and Yannis Haralambous: Practical LATEX and Acrobat — how you can get rich PDF documents automatically from LATEX;
Michel Goossens: Practical `latex2html`, serious strategies for converting large document databases;
Bart Wage: Real-world electronic journals — how Elsevier Science approaches on-line journal creation; members of the *HyperTEX Project* will describe their approach to publishing academic papers.

We are organizing presentations by various publishers on how they are using TEX, and how they solve practical problems. We will have practical workshops to discuss issues of general concern:

- publishing in physics;
- using Scientific Word and Maple;
- linguistics;
- virtual fonts;
- ScholarTEX and the humanities;
- converting SGML to LATEX;
- book design with TEX.

There will be a special session on TEX standards; the working groups of the TUG Technical Council will be invited to report on their progress. A particular feature will be the presentation of the final report of the TEX Directory Structure (TDS) working group, specifying how to set up TEX systems, and a discussion of organising TEX distributions and archives. **Norman Walsh** (O'Reilly), the coordinator of the TDS effort, and **Rich Morin** (Prime Time Freeware), the publisher of the first CTAN CD-ROM, will act as moderators.

Thursday will end with the TEX Users Group business meeting, your chance to hear what the Board of Directors has been up to over the year, check the financial accounts, and put forward your suggestions.

↪ **Friday** ↩

Projects: TEX in practice involves more than just quality typesetting; we can also delve into TEX tools, TEX programming style, the lessons we have learnt from TEX, and TEX as a programming language. Papers will include:

Sergey Lesenko: T1Part (Type1 partial);
Alan Hoenig: Poetica: a virtual font project;
Włodek Bzyl: Literate `plain.tex` source is available!
Bart Childs and Deborah Dunn: Teaching CS/1 courses in a literate manner;
Matthew Swift: Modularity in LATEX;
Sebastian Rahtz and Michel Goossens: Another look at LATEX-to-SGML conversion.

Reisverslag BachoTEX'95

Erik Frambach

e.h.m.frambach@eco.rug.nl

III Ogólnopolska Konferencja TEX-owa, BachoTEX'95

De Poolse TEX-gebruikersgroep 'GUST' is een actieve club met zo'n 120 leden, die eenmaal per jaar een meerdaagse bijeenkomst organiseert. Dit jaar werd die bijeenkomst voor de 3e keer gehouden in het plaatsje Bachotek, dat voor die gelegenheid wordt omgedoopt tot BachoTEX. De bijeenkomst start traditioneel op vrijdagavond en duurde dit jaar tot en met woensdagochtend. Met z'n vijftigen zaten we in een soort bungalowpark aan een groot meer midden in de bossen.

GUST
Grupa
Użytkowników
Systemu
TEX

GUST maakt er een goede gewoonte van buitenlandse gasten uit te nodigen om lezingen te geven. Dit jaar waren Wietse Dol en ik uitgenodigd om over 4allTEX te spreken, Kees van der Laan om over BLUe zaken te te spreken, en verschillende andere internationaal bekende figuren, van wie de meesten helaas niet konden komen. Desalniettemin deed de conferentie sterk denken aan EuroTEX'94 in Gdańsk, niet in de laatste plaats door de opvallend hoge kwaliteit van de presentaties.

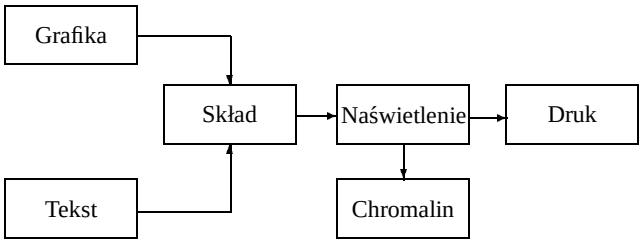
Op zaterdag werd de conferentie geopend door de voorzitster van GUST, Hanna Kołodziejska. De eerste presentatie was van Marek Ryćko die uitvoerig beschreef hoe TEX tewerk gaat bij het verwerken van invoer. Veel aandacht werd daarbij besteed aan de betekenis van TEX's 'main vertical list' en wat daar allemaal mee gebeurt bij het maken van alinea's en pagina's.

Vervolgens vertelde Norbert Jankowski over de principes van het conversieprogramma 'LATEX2HTML'.

's Middags was het Kees van der Laans beurt om zijn BLUe format nader toe te lichten. Daarna legde Stanisław Wawrykiewicz uit hoe de TEX-archieven op de file servers in Polen zijn opgezet. 's Avonds werd er een kampvuur aangelegd waarin worstjes werden geroosterd. 'TEX-bonding' in een heel ontspannen vriendelijke sfeer.

Zondagochtend weidde Andrzej Tomaszewski ons in in de kunst van de microtypografie. Aan de hand van vele voorbeelden liet hij zien wat de effecten zijn van ligaturen, kerning, sierletters en ander fraais. Jammer dat de lezing puur Pools was en hij zelf ook geen Engels spreekt.

Na de koffiepauze legden Agnieszka Polak en Bogusław Lichoński uit wat erbij komt kijken om in TEX met kleuren te werken.



Phil Taylor (UK) wees ons 's middags op de verschillen tussen `\csname x\endcsname` en `\def\x`. De eerste krijgt als defaultwaarde `\relax` terwijl de tweede als default `\undefined` krijgt. Deze eigenschappen zijn verwarrend maar kunnen ook toegepast worden om elegantere macro's te schrijven. Verder vertelde Phil over de ongeschiktheid van TEX als markup-language. Phil meent dat TEX-gebruik kennis op een te laag niveau vereist, wat ondermeer blijkt uit de noodzaak om op de juiste plekken `{}` of `\ ` toe te voegen om te voorkomen dat TEX spaties wegsnoept. Als alternatief stelt hij een SGML-achtige markup voor die geheel in TEX geïmplementeerd is, en bovengenoemde bezwaren niet kent.

De hele avond was voor GUST-leden geweid aan de vraag 'Quo vadis GUST?'. Daar bleek men vele uren over te kunnen en willen discussiëren.

Maandagochtend vertelde Piotr Pianowski over de toepassing van kleur in TEX via PostScript. Jammer dat Piotr's presentatie in het Pools was en hij zelf moeizaam Engels spreekt, want hij heeft enorm veel kennis in huis. Enkele TEX/PostScript macro's van zijn hand hebben we dankbaar gekopieerd.

Verder mocht ondergetekende die ochtend vertellen over 4allTEX en demonstreren wat de gelijknamige cd-rom allemaal in huis heeft. De middag was gereserveerd voor bestuursverkiezingen van GUST. Een groot deel van de avond bleek ook nog nodig om een en ander af te ronden. Desondanks werd het erg gezellig met gitaarmuziek, vele moppen en enkele goocheltrucs.

Dinsdagochtend vertelde Krzysztof Leszczyński over WEB. Geheel in het Pools en zonder plaatjes of voorbeelden, dus weinig leerzaam voor mij. Włodek Bzyl demonstreerde vervolgens hoe TEX zelf gebruikt kan worden voor Literate Programming aan de hand van documentatie bij `plain.tex`.

Polen heeft enkele grote Metafont-specialisten in huis, van wie Bogusław Jackowski wellicht de exponent is. Hij liet zien hoe met eenvoudige middelen Metafont gebruikt kan worden om tekeningen te maken die vervolgens en EPS-formaat worden weggeschreven. De PostScript-code wordt in de log file geschreven en daar later weer uit gefilterd met bijvoorbeeld een AWK script.

Bogusław Jackowski, getekend door Toni Walter

's Middags legde Lutz Brikham (DE) uit hoe zijn TEX debugger werkt. Dit systeem maakt het mogelijk om TEX-code op heel laag niveau te onderzoeken. Alle bekende features zoals een 'breakpoint manager', een 'macro browser', een 'variable inspector' en een 'token trace window' zitten erin. De hele interface is gebaseerd op X-Windows en wordt aangestuurd door Tcl (Tool Command Language).

Julita Bolland vertelde 's middags over de structuur van Chinese karakters. Dat deed ze in wat zij noemt 'Penglish': eerst uitleg in het Pools, vervolgens een korte Engelse vertaling. Een heel creatieve oplossing voor het taalprobleem.

Daarna liet Jerzy Ludwichowski de mogelijkheden zien van de file server 'Alex' in Torun. Via een modem kan contact gemaakt worden en is de hele wereld via NFS (Network File System) bereikbaar.

Woensdag lichtte Phil Taylor een tipje van de sluier op van 'ε-TEX, een 100% compatibele opvolger van TEX', zoals hij het noemt. Dit systeem kent naast de bekende 'TEX-mode' een 'extended mode' waarin enkele nieuwe commando's beschikbaar zijn die het leven van de TEX-gebruiker kunnen vereenvoudigen en/of extra functionaliteit bieden. Net als 'Omega' en de TEX debugger is ε-TEX gebaseerd op 'change files' op de originele WEB-code.

Als afsluiting vertelde Mariusz Czerniak over de structuur van LATEX2$_\varepsilon$. Om twaalf uur sloot de nieuwe GUST-voorzitter Tomek Przechlewski de bijeenkomst officieel af.

Terugblikkend op de hele conferentie en alles eromheen moet ik zeggen dat ik onder de indruk ben van het niveau waarop TEX en Metafont in Polen worden bedreven. Hier wordt op fundamenteel vlak aan de weg getimmerd, ofschoon de Polen daar zelf heel nuchter en bescheiden over spreken. Dat siert hun alleen maar. Ik hoop en verwacht in de nabije toekomst nog veel fraais uit het oosten.

EuroTEX'95

September 4th – September 8th, 1995

Papendal, Arnhem

↪ Call for papers ↩

The EuroTEX conference 1995, including tutorials, will take place from:

September 4th until September 8th

in the Netherlands. The conference will be held at **Papendal**, near the city of Arnhem.

Papendal is located in one of the most beautiful areas of the Netherlands. Right in the middle of the vast woods of the province of Gelderland. About eight kilometers west of Arnhem. Tucked away under the lee of the green Veluwe-fringe.

The conference starts on September 4th in the afternoon and runs until September 7th noon.
Thursday afternoon and Friday September 8th are reserved for courses and/or tutorials.

The theme of the conference is: **The TEX Toolbox**

Wherever we talk about 'TEX' we mean TEX and a format, such as plain, LATEX, AMSTEX, etc.

We would like to have talks that deal with topics such as:

- the integration of TEX in the working environment, or using TEX in full (bidirectional) co-operation with all kinds of other applications;
- conversion to and from TEX;
- dissemination of typeset material (PostScript, Acrobat);
- importing text and/or data from other applications (spreadsheets, databases) into TEX documents;
- how to add value to documents by using TEX, so that they can not only be easily typeset, but can also be turned into 'active' hyperdocuments, by translating them into SGML/HTML or Acrobat, or be used as data for database searches;
- multi-discipline, multi-language aspects of TEX;
- 'novel' uses of TEX such as bi/multi-lingual translations/dictionnaries, legal texts, poetry,. . .

We would also like to offer courses and/or tutorials on any of the following topics:

- introduction to using LATEX;
- LATEX for package and class file writers;
- making PostScript (Truetype) fonts available for use in TEX documents;
- database publishing with (LA)TEX;
- (LA)TEX and hypertexts, HTML, SGML, hyperTEX.

If you feel qualified to teach one of these subjects or think other topics should be covered, please send us your proposal.

Please notice the following (preliminary) deadlines:

April 10th : date when abstracts should be received
April 20th : date to tell authors their abstract is
: accepted/rejected
May 15th : date to receive full papers (using LATEX or
: EuroTEX style)
June 15th : date the reviews should be finished
June 25th : date to receive the final paper

Your abstracts and proposals can be sent to `eurotex-papers@cs.ruu.nl`

If you have no access to e-mail you can send your contribution to:

EuroTEX '95 program committee
Kooienswater 62
2715 AJ Zoetermeer
The Netherlands

In such a case we would ask you, to submit your paper in electronic form on a (DOS) floppy disk.

– – ∗ – –

↪ Dringend vrijwilligers gevraagd! ↩

We zoeken nog meer vrijwilligers om de organisatie van de conferentie tot een sukses te maken. Zet alle aarzelingen opzij, bekijk het lijstje en meld u aan voor een bijdrage die u ligt.

EuroTEX **moet** een succes worden, en heeft daarvoor ook uw hulp **nodig**!

De volgende taken wachten op een of enkele vrijwilligers:

- werving sponsors (bedrijven of instellingen die met TEX, LATEX of electronisch publiceren te maken hebben, of om andere redenen financiële of materiële steun willen geven).
- secretariaat, administratie van de inschrijvingen;
- beheer van de financiën van de conferentie, al dan niet samen met het secretariaat;
- samen met de MAPS-redaktie samenstellen en produceren van de *Proceedings* van de conferentie;
- contactpersoon voor het conferentieoord Papendal;
- beheer van het *Bursary Fund*, het fonds ter financiële ondersteuning van potentiële deelnemers voor wie de kosten te hoog zijn;
- organisatie van een **Oost-Europa bus**, gezamenlijk vervoer vanuit Oost-Europa ter verlaging van de reiskosten;
- organisatie van het *Social Event*, inclusief diner en eventueel vervoer.

Neem een stukje van de organisatie op je schouders en meld u aan!

per e-mail: `eurotex@cs.ruu.nl`

per telefoon: Erik Frambach, 050-633785 (werk), 050-421826 (privé)

Simon Pepping, `s.pepping@elsevier.nl`
Erik Frambach, `e.h.m.frambach@eco.rug.nl`
Piet van Oostrum, `piet@cs.ruu.nl`

– – ∗ – –