Nederlandstalige TEX Gebruikersgroep

Aan	:	Leden Nederlandstalige TEX Gebruikersgroep,
	:	Secretariaten Internationale TEX Gebruikersgroepen.
Betreft	:	Verslag 9^e NTG bijeenkomst
Locatie	:	Centrum voor Wiskunde en Informatica te Amsterdam
Datum	:	4 juni 1992
Behandeling	:	bij de volgende bijeenkomst (19 november 1992 te Meppel)

Agenda:

Bijlagen:

De NTG vereniging

Voorzitter:	C.G. van der Laan, Internet: cgl@rug.nl
Secretaris:	G.J.H. van Nes, ECN, Service Unit Informatica, Petten. Internet: vannes@ecn.nl
Penningmeester:	J.L. Braams, PTT Dr Neher Laboratorium, Leidschendam. Internet: j.l.braams@research.ptt.nl
Bestuursleden:	T.A. Jurriens, RUG, Groningen. Internet: taj@astro.rug.nl J.J. Winnink. Internet: winnink@ecn.nl
Postadres:	Nederlandstalige TEX Gebruikersgroep, Postbus 394, 1740 AJ Schagen.
Postgiro:	1306238, t.n.v. Penningmeester NTG, Zoetermeer.
Internet:	ntg@hearn

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, symposia en tentoonstellingen m.b.t. TEX en 'TEX-produkten', en door het onderzoeken en vergelijken van TEX met soortgelijke/aanverwante produkten, b.v. SGML.

De **NTG** biedt haar leden ondermeer het volgende:

- Tweemaal per jaar een **NTG**-bijeenkomst.
- Tweemaal per jaar de uitgebreide NTG MAPS (Minutes and APpendiceS).
- Indien mogelijk eenmaal per jaar open '**NTG**-dagen', waar naast lezingen, ook cursussen (speciaal tarief voor leden) worden gegeven.
- 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 een volledige MS-DOS versie van TEX, LATEX, en een previewer.
- De discussielijst TEX-NL waarop vragen gesteld worden. Ook worden er via deze listserver ervaringen uitgewisseld.
- Aktiviteiten in werkgroepen.
- Korting op (buitenlandse) TEX congressen en cursussen en op het lidmaatschap van TUG.
- Eenmaal per jaar een ledenlijst met per lid informatie welke software en welke hardware, in relatie met TEX, wordt gebruikt.

Lid worden kan door overmaking aan de penningmeester van het verschuldigde contributie bedrag. Daarnaast dient een informatieformulier te worden ingevuld, welke laatste via het secretariaat te verkrijgen is.

De contributie voor een persoonlijk lidmaatschap bedraagt f 75,–, de contributie voor een instituutslidmaatschap bedraagt f 200,–. 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 50,– per persoon.
Voor studenten geldt eveneens een tarief van f 50,– (geen stemrecht). Voor afwijkende regelingen dient contact met het bestuur opgenomen te worden.
Een gecombineerd NTG/TUG lidmaatschap bedraagt f 170,- per jaar (i.p.v. f 75,- + $ 60).

De statuten van de Nederlandstalige TEX Gebruikersgroep zijn via het secretariaat of via de fileserver te verkrijgen.

Nederlandstalige TEX Gebruikersgroep

Aanwezig : A. Al-Dhahir (UT); P.W. Bison; P. Bloemen (TUE); J. Braams (PTT); L. de Coninck (de Kraal); N. Cox (KUN); M. Dings (AKZO); W. Dol (RUG); C.M. Fortuin (Hogeschool Gelderland); E. Frambach (RUG); F. Goddijn; S.Y.A. Han; H. Jurriens (UT); T.A. Jurriens (RUG); R. Kappert (KUN); J. Krugers (Techn. Marketing Cons.); G.D.C. Kuiken (Kuiken); C.G. van der Laan (RUG); A. de Leeuw van Weenen (RUL/VTW); H.A.N. van Maanen; H. van der Meer (UvA); H.P.A. Mulders (KUB); G.J.H. van Nes (ECN); N.A.F.M. Poppelier (Elsevier); D. Salomon (California State University); R. Smedinga (RUG/Inf.); W. Smit; P. van Summeren (Scan Laser); N. Temme (CWI); E.J. Vens (RUG/ICCE); J.E. van Weerden (RUU); F. van de Wiel (CWI); J.J. Winnink; R.E. Youngen (AMS);

Notulisten : Gerard van Nes & Jos Winnink

1 Opening

Om ongeveer 10:15 uur heet voorzitter Kees van der Laan een ieder van harte welkom en opent de bijeenkomst. In het bijzonder heet hij David Salomon en Ralph Youngen welkom, die op deze bijeenkomst als gastsprekers zullen optreden. David Salomon is tevens de docent van de cursus: *'NTG's Advanced TEX course: Insights & Hindsights'* die van 15 tot 19 juni zal worden gehouden in Groningen. Voor een beperkt aantal geïnteresseerden is het mogelijk om de syllabus van deze cursus, die ongeveer 250 pagina's omvat, tijdens deze bijeenkomst tegen kostprijs aan te schaffen.[1]

De reden dat wij te gast zijn bij het CWI is een gevolg van het feit dat de contacten met de wiskundigen zijn verstevigd.

Tijdens het middaggedeelte van deze bijeenkomst zullen er verschillende voordrachten worden gehouden met als thema *'TEX and Scientific Publishing'*.

Voor het najaar van 1992 wordt een bijeenkomst voorbereid waarbij als thema gedacht wordt aan *'The future of TEX/LATEX'*. Ook de ontwikkelingen met betrekking tot het LATEX 3.0 project zullen dan aan de orde komen.

Gewezen wordt op het aanwezige uitgebreide materiaal op de leestafels, en de mogelijkheid om TEX boeken aan te schaffen bij Addison-Wesley met een korting van 10%.[2]

2 Verslag bijeenkomst 21 november 1991

Bij dit verslag zijn geen opmerkingen en aldus wordt het goedgekeurd.

Gerard van Nes noemt de problemen rond de versturing van het verslag naar het grote aantal (over de honderd) NTG leden tegelijkertijd via het netwerk waarbij verschillende abonnees het materiaal in veelvoud in hun e-mail box hadden ontvangen. De fout lag bij het netwerk zelf en is hopelijk bij de volgende verzending opgelost.

Naar aanleiding van het verslag wordt het volgende opgemerkt:

- **Werkgroep 13: De IJ-ligatuur**
  De situatie is wat blijven zweven. Erik Jan Vens deelt mee dat er een plaatsje is ingeruimd voor de ij-ligatuur in de DC-fonts.
- **Picture Environment**
  Hans van der Meer heeft de door hem ontwikkelde 'picture environment' naar Johannes Braams gestuurd. Deze heeft nog geen gelegenheid gehad om het op de fileserver te zetten maar zal dat spoedig doen. Deze picture environment zal ook worden meegenomen in het kader van het LATEX 3.0 project.
- **Ledenbestand**
  Het aantal leden is inmiddels gegroeid van 150 naar 180. De vereniging groeit op dit moment gemiddeld met ongeveer 45 leden per jaar.
- **Bijlage Q**
  Joachim Schrod heeft medegedeeld dat het artikel *'The Components of TEX'*, zoals dat in de MAPS 92.1 als bijlage Q is gepubliceerd, een versie is waar hij niet meer achter staat. Mede naar aanleiding van discussie's met anderen is hij bezig een vernieuwde versie te maken. De redactie was hiervan op de hoogte, maar meende dat het bij het niet

Editor van deze MAPS is: G.J.H. van Nes.
Het verslag van de NTG bijeenkomst op 4 juni 1992 is (in concept) eind september 1992 via e-mail dan wel via de post reeds gestuurd naar alle NTG leden.

[1] Het cursusmateriaal is nog beperkt aanwezig. De prijs is, inclusief portokosten (binnen Nederland/België): *f* 30,– voor NTG leden en *f* 40,– voor niet NTG leden. Bestelling via de penningmeester Johannes Braams.

[2] Addison-Wesley was aan het begin van de middag aanwezig met een uitgebreide verzameling TEX/LATEX/PostScript boeken.

beschikbaar zijn van de vernieuwde versie, de oude nog wel voldoende waarde had om op te nemen in de MAPS. Wanneer de nieuwe versie beschikbaar komt zal deze zeker worden gepubliceerd.

3 Ingekomen stukken en Mededelingen

De volgende mededelingen worden gedaan:

- Berichten van verhindering zijn ontvangen van de leden Biegstraaten, Heslinga en Vanoverbeke.
- Ontvangen zijn diverse tijdschriften van ondermeer de verschillende TEX zustergebruikersgroepen, TTN en SGML.
- In een tijdschrift gericht op de Atari ST gebruikers is een aardig artikel over TEX verschenen.
- Sinds de vorige vergadering is er contact geweest met het Wiskundig Genootschap. Dit contact heeft ondermeer geresulteerd in publikatieruimte voor de NTG in *'Mededelingen van het Wiskundig Genootschap'*. Auteur Kees van der Laan.
- Er hebben zich geen kandidaten voor één van de opengevallen plaatsen in het bestuur aangemeld.
- De NTG heeft een donatie van *f* 1000,– gedaan aan het LATEX 3.0 project.
- Bij TUG wordt met veel belangstelling de MAPS gelezen.
- Voor de cursus van David Salomon zijn op dit moment ruim 40 deelnemers genoteerd. Dit aantal zal mogelijk nog wat stijgen.
- In 1993 bestaat de NTG vijf jaar. Er wordt geprobeerd om ter gelegenheid van dit lustrum een congres te organiseren. Op dit moment hebben twee mensen zich bereid verklaard te zaak te willen trekken. Er zal worden nagedacht of het lustrum te combineren is met EuroTEX 93, maar gezien de beschikbare tijd rijzen hier wat vraagtekens. Meer informatie zal spoedig volgen.
- Het electronische tijdschrift TeXHaX, welke de laatste tijd erg onregelmatig verscheen, wordt nu door Peter Abbott vanuit Engeland verzorgd. Overigens kan deze nieuwe TeXHaX distributie problemen opleveren aangezien de gateway die de verbinding verzorgt tussen het Engelse computernetwerk en de rest van de wereld alles behalve betrouwbaar is.
  Het electronische tijdschrift TeXMaG is ter ziele. Daarentegen verschijnt UKTeX wel regelmatig. Ook de 'usenet newsgroup' is zeer actief.

4 NTG jaarvergadering

- **Algemeen**
  Theo Jurriens vindt de opkomst voor deze bijeenkomst wat laag. Hij vraagt zich af of het niet mogelijk is om *alle* bestuurszaken op één bijeenkomst te concentreren en de andere bijeenkomst te gebruiken voor 'nuttiger' zaken.
  Het antwoord is dat er dan een probleem is i.v.m. de statuten. In de praktijk wordt alleen nog de begroting op de najaarsbijeenkomst behandeld. Geopperd wordt, dat mogelijk bij deze vergadering in Amsterdam, een aantal leden in het westen van het land de bestuurszaken overslaan en pas in de middag zullen komen. Een andere reden zou kunnen zijn dat de NTG haar taak steeds meer volbrengt en TEX steeds meer ingeburgerd raakt, waardoor leden de indruk hebben dat het allemaal wel goed gaat zo. Opgemerkt wordt dat het ledental een gestage groei te zien geeft zoals reeds eerder op deze vergadering is opgemerkt.
- **Jaarverslag van de secretaris**
  Er zijn geen opmerkingen bij het jaarverslag van de secretaris.
- **Jaarverslag van de penningmeester**
  Er zijn geen opmerkingen bij het jaarverslag van de penningmeester.
- **Royementen**
  Er zijn leden die het opzeggen van het lidmaatschap van een vereniging regelen door het niet betalen van de contributie. Zij doen dit in plaats van een gebruikelijke schriftelijke opzegging.
  Het bestuur stelt voor om de heer Grootenhuis formeel te royeren wegens het niet nakomen van zijn financiële verplichtingen jegens de NTG ondanks herhaaldelijke verzoeken van het bestuur.
  De vergadering gaat met dit voorstel akkoord. De penningmeester zal het betreffende lid hiervan schriftelijk in kennis stellen.
- **Verslag van de kascontrolecommissie**
  Het verslag van de kascontrolecommissie wordt goedgekeurd. De penningmeester wordt gedecharcheerd voor het gevoerde financiële beleid.
  Ton Biegstraaten treedt af als lid van de kascontrolecommissie. Andrea de Leeuw van Weenen blijft zitten. Jules van Weerden wordt verkozen tot nieuw lid.
- **Bestuursverkiezingen**
  Gerard van Nes en Huub Mulders treden af. Gerard van Nes stelt zich herkiesbaar; Huub Mulders niet. Het bestuur stelt als kandidaat voor de opengevallen plaats Theo Jurriens voor. Er hebben zich geen andere kandidaten gemeld.
  Bij acclamatie worden zowel Gerard van Nes als Theo Jurriens verkozen door de vergadering. Huub Mulders wordt bedankt voor de activiteiten die hij voor de NTG in de afgelopen jaren heeft verricht.

5 Verslag/discussie werkgroepen

Erik Jan Vens meldt dat hij zich terugtrekt uit alle werkgroepen waarin hij zitting had. Dat zijn de werkgroepen 4, 7 en 15.

Frans Goddijn vindt opmerking op pagina 58 van MAPS 92.1: *'ik vond Corrie en Norma aardig en realistisch,'* een leuke opmerking maar kon hem niet zo goed plaatsen.

5.1 Werkgroep 15: Future of TEX/LATEX

Johannes Braams maakt melding van de activiteiten op dit gebied.

- **LATEX 3.0**
  Op dit moment wordt er hard aan het LATEX 3.0 projekt gewerkt. Ondanks het feit dat er gedurende een tijd geen informatie naar buiten kwam, zijn de werkzaamheden doorgegaan. Op dit moment bestaat er een β-versie van de nieuwe kernel van LATEX 3.0. Echter er is nog niets officieel vrijgegeven. Volgens het tijdschema is er in de zomer van 1993 een β-versie van LATEX 3.0 beschikbaar.
  Frank Mittelbach zal mogelijk op de komende NTG najaarsbijeenkomst nader op het LATEX 3.0 project ingaan.
- **Future of TEX**
  Op initiatief van DANTE is er een discussielijst opgezet onder leiding van Joachim Schrod om te discussiëren over NTS (New Typesetting System). Het doel is om te komen tot een formulering van eisen voor een opvolger van TEX. Mogelijke realisatie pas na 5 à 10 jaar. Dit alles los van de werkzaamheden van Frank Mitelbach.
  Er zijn op dit moment twee stromingen:
  - TEX 3 + verbeteringen/uitbreidingen;
  - Een kompleet nieuw systeem dat van de grond af aan opgebouwd is, gebruik makend van de ervaringen met TEX.

  Nelson Beebe schreef in TUGboat, Volume 13.1, p. 6, dat Knuth van mening is dat TEX wel mag worden uitgebouwd maar dat het dan niet meer de naam TEX mag hebben. Een naam als ETEX of iets dergelijks is echter geen enkel probleem. Ook het gebruik van (delen van) de *source* van TEX 3 is toegestaan voor een dergelijk project.
  Het probleem is nu om consensus te bereiken over de toekomstige ontwikkelingen en wildgroei te voorkomen. Mogelijk kan de Board of Directors van TUG hier een rol in spelen.
  Gediscussieerd wordt verder over het al dan niet compatible zijn van komende (van TEX afgeleide) nieuwe TEX-achtige 'standaarden' en de ontwikkelingen die buiten TEX gaande zijn doch wel in de toekomst mogelijk door TEX te gebruiken zijn (ondermeer bij SGML, Adobe, Rank Xerox; echter niet altijd onder de Public Domain noemer!).
- **Miscellaneous**
  Een korte 'levendige' (en soms ongecontroleerde) discussie vindt plaats over een aantal WordPerfect zaken (formule editor welke feitelijk UNIX 'eqn' is inclusief de problemen, wel ruim voldoende voor de markt waarvoor WP bedoeld is; de zeer beperkte referentie/bibliografie mogelijkheden; de geheel andere leercurve). Het grote geld is op dit moment echter te verdienen met de verwerking van platte tekst, dus met de bekende al dan niet perfekte *woordverwerkers*, aldus enkele aanwezigen.
  Willem Smit merkt op dat LATEX ook heel goed bruikbaar is voor andere dingen dan het zetten van wiskundige formules. Zeker ook voor de platte tekst zelf. Het gebrek aan een groot scala van leerboeken en help-faciliteiten, zoals wel bij andere tekstsoftware beschikbaar is, is voor hem zeker geen probleem.
  Een artikel van Huub Mulders m.b.t. formule editors in TEX, WP en DecWrite wordt genoemd.
  Marcel Dings vindt dat het ontwerpen van huisstijlen met LATEX moeilijker is dan met een DTP-pakket en heeft ondermeer om deze redenen dan ook zijn toevlucht moeten nemen tot een DTP-pakket voor het ontwikkelen van een huisstijl.
  Jeroen van Maanen zou graag meer flexibele LATEX stijlen gezien hebben. Johannes Braams antwoordt hierop dat met de komst van LATEX 3.0, de documentatie sterk verbeterd wordt (nieuw LATEX boek) en met name dat documentatie over het zelf ontwikkelen van stijlen zal worden opgenomen. In de huidige versie van LATEX ontbreekt dit laatste.
  N.a.v. de opmerking van Nico Poppelier dat je stijlen niet moet aanpassen, merkt van Maanen op dat hij eigenlijk een systeem wil hebben waarmee een typografisch ontwerper op een eenvoudige wijze ('keuzeschakelaars') een huisstijl kan maken.
  Nico Poppelier merkt vervolgens op dat Victor Eijkhout het LOLLIPOP formaat heeft ontwikkeld waarmee zonder veel moeite stijlen kunnen worden gegenereerd.

5.2 Werkgroep 7: PC-zaken

Jos Winnink deelt mede dat de organisatie rond de emTEX distributie nog niet geheel af is. In principe zijn alle modulen aanwezig. Het installatiescript van Piet Tutelaers moet nog wat aangepast worden. Behoefte is er aan een script voor volledige en voor een beperkte installatie.

Discussie ontstaat er over de manier van verspreiding van de emTEX distributie. Er wordt ondermeer op gewezen dat het mogelijk is om via de Public Domain Software Service van de HCC, programmatuur te verspreiden. Dus waarom ook niet de (komende) NTG distributie van emTEX? Wel zal het up-to-date houden dan een probleem worden. Daarnaast zou de verspreiding via het NTG secretariaat kunnen, waarbij moet worden uitgegaan dat de diskettes door derden worden vermenigvuldigd.

Alaaddin Al-Dhahir vraagt of het niet mogelijk is om regionale vertegenwoordigers van de NTG als steunpunt aan te wijzen. Frans Goddijn vindt dat de service professioneler zou moeten. Gedacht zou ook kunnen worden aan een soort *'helpdesk'* en/of een *'06-nummer'*. Algemene mening: *een goed service apparaat is belangrijk.*
De voorzitter antwoordt dat alle werkzaamheden van de NTG vallen in de categorie 'liefdewerk en oud papier'. Het gevolg hiervan is dat de NTG voorzichtig

moeten omspringen met de beschikbare menskracht. Maar mogelijk zijn de aangegeven kanalen zoals HCC, Universiteiten, Studenten Verenigingen en SURF goed bruikbaar. Mogelijk ook CWI.
Voorlopig richt de NTG zich in eerste instantie op specifieke groepen gebruikers (wetenschappers) en ook op SURF. Nico Poppelier wil de contacten met SURF wel leggen (dan wel medewerking hieraan verlenen).

Alaaddin Al-Dhahir zegt dat hij van de server in Utrecht een incomplete MS-DOS versie van DVIPS heeft gehaald. Jos Winnink zegt toe dit probleem met Alaaddin te zullen oplossen. Op de server in Utrecht staat namelijk een complete versie aangevuld met een versie die slechts de voor MS-DOS benodigde executables bevat.

Genoemd wordt dat op de server in Eindhoven DVIPS versie 5.483 beschikbaar staat.

Jules van Weerden meldt dat via ACCU/computershop op dit moment ook versies van TEX verspreid worden. Hier zou ook de door NTG samengestelde emTEX versie gedistribueerd kunnen worden.

6 Rondvraag

- **Willem Smit** vertelt dat hij een versie van RevTEX heeft en het aan geïnteresseerden beschikbaar wil stellen. RevTEX is een macro pakket voor natuurkundigen dat wordt gebruikt bij produktie van de *'Physical Review Letters'*, Poppelier meldt dat de software ook via een of meer fileservers te verkrijgen is.
- **J. Krugers** heeft een (MS Windows 3.0) versie van het WYSIWYG pakket *'Scientific Word'* dat gebruikt maakt van TEX/LATEX (kan TEX/LATEX lezen en genereren). Het pakket is echter afgestemd op de typografie die in de Verenigde Staten gebruikelijk is. Hij zou graag advies willen krijgen om het pakket zodanig te doen wijzigen dat het voldoet aan de typografische normen in Europa. Vanwege de geheel nieuwe gebruikersinterface zou het pakket voor TEX/LATEX gebruikers wel eens interessant kunnen zijn.
  Geantwoord wordt, dat mogelijk binnen de NTG mensen zijn die het pakket willen reviewen en van (schriftelijk) commentaar willen voorzien. Eerder ontvangen commentaar van twee Nederlandse testsites blijken door de Amerikaanse fabrikant geheel in de nieuwe versie opgenomen te zijn. Belangstellenden kunnen direct met hem contact opnemen (voor adres zie NTG ledenlijst).
- **Alaaddin Al-Dhahir** vraagt hoe het zit met het gecombineerde NTG/TUG lidmaatschap voor instituutsleden.
  Johannes Braams antwoordt hierop dat dit met TUG moet worden overlegd omdat er wat verschillen zijn in opzet tussen instituutsleden bij NTG en bij TUG. Wel is het mogelijk om het TUG lidmaatschap via de NTG te voldoen.
  In aanvulling hierop deelt de penningmeester mede dat op dit moment ongeveer 30 NTG/TUG lidmaatschappen geregistreerd zijn, hetgeen ruim boven de verwachtingen van TUG ligt.

7 NTG presentaties 'TEX and Scientific Publishing'

Het middagprogramma bestond uit de volgende voordrachten:

- *'From Observation to Publication'*, door Theo Jurriens. Hij besprak de situatie met betrekking tot het publiceren m.b.v. TEX/LATEX in de wereld van de astronomie (zie bijlage in MAPS 92.1 en het boek 'Desktop Publishing in Astronomy & Space Sciences', van André Heck, uitgever World Scientific).
- *'On Standard Notations in Mathematics'*, door Nico Poppelier. Hij vroeg ondermeer aandacht voor de problemen rond het 'gestandaardiseerd' opslaan en het uitwisselen van wiskundige formules in een context-afhankelijke omgeving (zie bijlage in MAPS 92.2).
- *'$\mathcal{AMS}$-TEX based production'*, door Ralph Youngen. Hij gaf een uiteenzetting met betrekking tot de situatie bij de 'American Mathematical Society' (zie bijlage in MAPS 92.2).
- *'Math into BLUes'*, door Kees van der Laan. Over de problemen die kunnen ontstaan als wiskunde typografisch moet worden verwoord. Oplossingen werden aangedragen (zie bijlage in MAPS 91.1).
- *'Automatic Index Generation'*, door David Salomon. Hij hield als voorbereiding op de door hem te geven NTG-cursus, een voordracht over het automatisch genereren van een index (zie bijlage in MAPS 92.2).

8 Sluiting

Om ongeveer 17:30 uur wordt de vergadering door de voorzitter gesloten. De gastheer, het CWI, wordt bedankt voor de genoten gastvrijheid. De aanwezigen en in het bijzonder de buitenlandse gasten worden bedankt voor hun bijdragen.

Een deel van de aanwezigen gaat het, inmiddels tot een traditie uitgegroeide, gezamenlijke diner genieten.
De volgende vergadering is op:

donderdag 19 november 1992

te Meppel. Het thema: 'The future of TEX/LATEX'.

1 TEX kalender 1993

10	jun	NTG (11 ^{e})	KNMI, de Bilt
26–29	jul	TUG '93	Aston, England

2 Glossary

2.1 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 Belgie
SibTUG : LUG Siberië
UKTUG : LUG Engeland
YUNUS : LUG Turkije (feitelijk een discussielijst)
GUST : LUG Polen

2.2 Bulletins/journals

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

2.3 Diversen

AMS : American Mathematical Society
BoD : Board of Directors
SGML : Standard Generalized Markup Language
ltxiii : LATEX 3.0

3 Discount boeken en software voor NTG leden

3.1 Aanschaf TEX boeken

De aanschaf van TEX boeken door NTG leden (*met korting*) tijdens afgelopen maanden mei en juni (zie de vorige MAPS) heeft, vanwege het grootte succes, nu wederom een (tijdelijk) vervolg!
Evenals dit voorjaar is nu ook weer met Addison-Wesley overeengekomen dat NTG leden een korting ontvangen van 10% bij aanschaf van TEX boeken bij deze uitgeverij. Wel worden de verzendkosten in rekening gebracht (ongeveer f 10,–). Daarentegen worden de Amerikaanse prijzen aangehouden, hetgeen ook een extra korting betekent voor de NTG leden!
De aanbieding geldt echter tot uiterlijk 31 december 1992.

De volgende voorwaarden gelden:

1. Bestelling direct bij Addison-Wesley *t.a.v. Rita Snaddon persoonlijk.*
   Voor bestellingen die niet via haar binnenkomen geldt bovenstaande korting *niet*!
2. Betaling *alleen per credit card* (andere mogelijkheden zijn er niet).
3. Vermelding NTG lidmaatschapsnummer (zie ledenlijst).

Behalve voor de TEX boeken, geldt deze regeling ook voor een aantal boeken over PostScript die door deze uitgever worden uitgegeven.
Meer informatie over de boeken is te vinden in de folder die bij deze MAPS is bijgesloten.

3.2 Aanschaf TEX cursusmateriaal

Er zijn nog enkele exemplaren verkrijgbaar van het TEX cursusmateriaal van David Salomon (zie MAPS 92.1, blz. 149). Het boekwerk is ruim 250 bladzijden dik en bevat zeer waardevol materiaal inclusief zeer veel (praktische) voorbeelden. Prijs voor NTG leden f 30,– inclusief porto (niet leden betalen f 40,–).
Bestellingen kunnen gedaan worden door overmaking van het verschuldigd bedrag op de postgiro (1306238) t.n.v. de penningmeester Johannes Braams te Zoetermeer.
Wel vanzelfsprekend 'zolang de voorraad strekt'.

3.3 Aanschaf software 'Scientific Word'

Met de leverancier 'Technical Marketing Consulting' te Steenbergen is overeengekomen dat NTG leden die het pakket Scientific Word (zie bijlage $\mathcal{D}$ in deze MAPS) *voor 31 december 1992* bestellen, een korting ontvangen van 25% (op de aanschafprijs van f 2000,–).
Uitgangspunt bij deze korting is mede het feit dat NTG leden geen direct beroep hoeven te doen op zaken die direct met TEX/LATEX te maken hebben.
Bij bestellingen dient behalve de naam, ook het NTG lidnummer opgegeven te worden (zie ledenlijst).
Adres/telefoonnummer van de leverancier is te vinden in de betreffende bijlage van deze MAPS.

4 MAPS 1993

Sluitingsdatum inleveren artikelen/bijlagen voor de MAPS van 1993 is:
1 april (MAPS 93.1) en 1 oktober (MAPS 93.2).

In MAPS 93.1 zullen ondermeer de volgende bijdragen worden opgenomen (onder voorbehoud):
'The Future of TeX', 'Tutorial Virtual Fonts', 'Large scale manual production using LATEX by a major Graphics Software Company', 'Ervaringen met het zetten van een boek', 'Arbitrary SGML DTD-s in TeX', 'When TeX and Metafont work together', 'Sorting in BLUe', 'AsTeX', 'TEX/LATEX on PC systems', en 'Boekbespreking LATEX Companion'.

Nadere richtlijnen voor auteurs zijn op te vragen bij de redactie.

Werkgroepen
Nederlandstalige TEX Gebruikersgroep

1. **Educatie:**
   A.W.W.M. Biegstraaten (TUD)
   M. Clark
   C.G. van der Laan *
   P. Tutelaers (TUE)

2. (werkgroep is vervallen)

3. **Evaluatie produkten (Ned. LATEX incl sty. files en afbreekregels; andere macrocollecties AMSTEX; converters K-talk; TEX naar ASCII; index programmatuur; dBase-TEX koppeling; adreslabels; verkrijgbaarheid etcetc.):**
   J.L. Braams (PTT Research Neher Lab) *
   M.A.J.H. Broeren (Océ Nederland B.V.)
   H.P.A. Mulders (KUB)

4. **Fonts (gebruik van Metafont):**
   H. Brouwer (EGD)
   A.J. de Meyer (RUU; Wiskunde)
   P. Tutelaers (TUE)
   F.J. Velthuis (RUG; Rekencentrum)
   J.C. de Moor (Theol. Univ.)
   J.J. Winnink

5. **Drivers, previewers, printers, postscript:**
   J.L. Braams (PTT Research Neher Lab)
   H. Brouwer (EGD)
   P. Tutelaers (TUE)

6. **Lijst en link met fotozetters:**
   G. Haayer (Styx Publications)
   T.A. Jurriens (RUG; Sterrenkunde)
   F.J. Velthuis (RUG; Rekencentrum) *

7. **PC-perikelen; campuslicentie etc.:**
   E. Algera (EGD; Amiga)
   A.H.A. Bloemen (TUE)
   G.J. Braas (EGD; Archimedes)
   H. Brouwer (EGD)
   P. Tutelaers (TUE)
   J.J. Winnink (-; DOS)
   E.B.J. van der Zalm (RUU; Atari)
   R. Veldhuyzen van Zanten (SARA; McIntosh)

8. **Nederlandse TEX gebruikersdag:**
   werkgroep (tijdelijk) op non-actief

9. **Integratie beelden en TEX:**
   H. Brouwer (EGD)
   T.A. Jurriens (RUG; Sterrenkunde) *

10. **SGML-TEX relatie:**
    A.W.W.M. Biegstraaten (TUD)
    D.C. Coleman (Elsevier Science Publishers)
    C.G. van der Laan
    N.A.F.M. Poppelier (Elsevier Science Publishers)

11. (werkgroep is vervallen)

12. **Beheerders handleiding/documentatie:**
    J.L. Braams (PTT Research Neher Lab)
    E.J. Evers (RUU; Geneeskunde) *

13. **Nederlandstalige TEX:**
    J.L. Braams (PTT Research Neher Lab)
    V. Eijkhout (Univ. of Illinois)
    D. van Leeuwen (RUL)
    N.A.F.M. Poppelier (Elsevier Science Publishers)

14. **Communicatie:**
    J.L. Braams (PTT Research Neher Lab) *
    V. Eijkhout (Univ. of Illinois)
    E.J. Evers (RUU; Geneeskunde)
    P. van Oostrum (RUU)

15. **TEX 3.0 (The Future of TEX):**
    H.P.A. Mulders (KUB)
    P. van Oostrum (RUU) *

* coördinator

Van uw MAPS Editor

Gerard van Nes

October 1992

1 Inleiding

Het editwerk van uw editor zit er weer op. Het kostte nog wel vele uurtjes zweten om alle binnengekomen, soms geheel verschillende bijdragen aan elkaar te knopen. Het is toch altijd weer een race tegen de tijd. Vandaar dat deze MAPS iets later verschijnt dan zou moeten, doch het is weer een document waar u ongetwijfeld waardevolle informatie uit kunt halen.
T.o.v. de eerste MAPS (toen werkelijk 'Minutes' met enkele APpendiceS; heeft u ze nog?), is er afgelopen jaren zeer veel gebeurd. De laatste serie MAPS zijn ook nauwelijks meer te vergelijken met de eerste nummers, zowel op kwalitatief als kwantitatief gebied. Alleen de verslagen van de NTG bijeenkomsten zijn redelijk stabiel gebleven. En de rest? De laatste twee à drie jaren rond de 150–160 pagina's per uitgave, boordevol TEX/LATEX en aanverwante zaken. MAPS editwerk is nauwelijks meer iets om 'even' tussendoor te doen.
In deze 'last minute' bijlage enkele zaken die ondermeer ook bij het samenstellen van deze MAPS naar voren kwamen.

2 De layout van de MAPS

De layout van ook deze MAPS is nauwelijks veranderd t.o.v. die van de laatste twee jaren. Af en toe is er wel wat bijgevijld, fouten zijn er echter ongetwijfeld nog steeds (het editorswerk moet ook weer niet exponentieel toenemen; wij zijn immers geen commerciële instelling).
In het verleden heeft er een discussie plaatsgevonden over mogelijke veranderingen in de typografie van de MAPS. Het onderwerp heeft vanzelfsprekend een zekere subjectiviteit en er zullen altijd wel voor- en tegenstanders blijven bestaan van elke soort typografie die voor de MAPS gebruikt wordt dan wel gebruikt zal gaan worden. Zaken als compactheid (zodoende tweekoloms uitvoering met bijbehorende corpsgrootte) en leesbaarheid (fontkeuze Times en bepaalde logische structuur afspraken) zijn met opzet door de (huidige) editor gekozen zoals ze nu zijn.[1] Opbouwende opmerkingen worden altijd ter harte genomen. *Kleine* aanpassingen zijn in de toekomst vanzelfsprekend altijd nog wel te verwachten. Zo zal bijvoorbeeld de volgende MAPS hoogstwaarschijnlijk niet meer op een 300dpi laserprinter worden gedrukt!
Duidelijk is afgeweken van ondermeer de bekende TUGboat typografie. Een min of meer *eigen gezicht* van de MAPS is belangrijk.

3 LATEX of plain TEX?

In MAPS 91.2 heeft u van de voorzitter een voorstel gelezen om te komen tot een afspraak hoe het materiaal van een auteur aangeleverd zou moeten worden. Genoemd werd daarbij de welbekende tugboat/ltugboat stijlen.
In eerste instantie is echter bewust gekozen om geen directe drempel te scheppen bij het aanbieden van materiaal voor de MAPS door een auteur. Veelal zijn artikelen reeds eerder geschreven, gebruikmakend van een bepaalde stijlfile. Omzetten door de auteur zelf zou een mogelijke terughoudendheid kunnen geven bij het aanbieden van materiaal. Daarentegen geeft het voor de editor natuurlijk wel wat extra werk.

Deze MAPS is geheel gemaakt via LATEX, ondanks het feit dat ongeveer 30% van het materiaal in plain TEX[2] werd aangeleverd. Zo zijn bijvoorbeeld de bijdragen over 'Shaded Rectangles with PostScript' en 'Typesetting Crosswords' ook geheel met LATEX te maken! Soms was bij het opnemen van de betreffende TEX code binnen de 'standaard' LATEX files van de MAPS wel een (relatief kleine) correctie (soms 'truc') noodzakelijk, *maar het kan*! Ook deze MAPS toont dus weer aan dat het opnemen van TEX code binnen een LATEX document in principe in het geheel geen probleem is.[3]
LATEX kan dikwijls meer dan men denkt.

Zoals mogelijk velen van u weten, is uw MAPS editor duidelijk geen voorstander van het direct gebruik van (plain) TEX zowel door de beginnende als ook door de gevorderde maker van documenten ('van notities t/m encyclopedieën'). Het gebruik van *een logische structuur* binnen een document is nu eenmaal *van essentieel belang* voor zowel de levensduur, de eventuele verdere verwerkingsmogelijkheden en het uniforme/consistente uiterlijk van een document, als voor *de manier van denken/werken* van de auteur.
Mede dankzij het werk van Leslie Lamport kreeg het TEX gebruik een geheel nieuwe dimensie in de vorm van de structurele LATEX schil. *Alles kan in principe m.b.v. LATEX*. Wel zijn er echter dikwijls aanvullende speciale macro's nodig (waarvoor er dan wel in plain TEX geprogrammeerd moet worden, doch dit dient feitelijk alleen te gebeuren door een kleine groep van TEX experts). Trouwens het samenstellen (feitelijk 'componeren') van een document m.b.v. een serie goede macro's hoeft in de huidige tijd van *Object-Oriented*

[1] Als iemand deze taak wil overnemen laat mij dat dan weten! Het kost wel wat tijd, doch men verkrijgt een brok ervaring!
[2] In bijna alle gevallen via electronic mail; één artikel inclusief eps files via diskettes, één via ftp.
[3] Alleen moet men wel goed beseffen wat men doet!

werken niet bepaald vreemd meer te klinken.
Het bovenstaande is ook de reden dat uw MAPS editor zich *ook* bij de samenstelling van deze MAPS, alleen maar gebruik heeft gemaakt van LaTeX *plus* een verzameling van benodigde TeX macro's.

4 Bijdragen MAPS

De bijdragen in de MAPS bestaan i.h.a. uit een mengeling van 'van alles wat'. Echter oplettende lezers zullen ongetwijfeld ontdekt hebben dat er altijd bijdragen bijzitten op het gebied van educatie, PostScript zaken, 'ongewone' TeX/LaTeX toepassingen, en algemene TeX/LaTeX beschouwingen: onderwerpen die de meeste lezers direct aangaan.
Daarnaast komt ook de (ver)gevorderde gebruiker aan zijn trekken met dikwijls de meest uiteenlopende macrobeschrijvingen.
Het zal opgevallen zijn dat veel bijdragen reeds dikwijls elders gepubliceerd zijn of later in een definitieve versie nog elders gepubliceerd zullen worden. Veelal is dat in het lijfblad van de TeX/LaTeXgebruiker/ster: *TUGboat*. De reden van deze 'dubbele publicatie' is i.h.a. *de waarde* die zo'n bijdrage heeft: bepaalde artikelen verdienen het eenmaal om zo breed als mogelijk verspreid te worden.
Ook bevatten de MAPS veelal bijdragen van eigen NTG leden die, vanwege het 'voor de vuist weg' karakter, niet direct elders gepubliceerd kunnen worden, doch wel een zodanige waarde bezitten dat veel lezers er zeer mee gebaat kunnen zijn.

5 Verwerking MAPS

Het verkrijgen van de diverse bijdragen is dikwijls toch altijd weer een kunst op zich. Veel auteurs sturen hun werk na een eerste verzoek keurig in. Daarentegen wordt ook soms het materiaal ruim na een sluitingsdatum ontvangen, zelfs enkele dagen voordat de hele boel naar de drukker[4] moet. Doch u zal ook wel ontdekt hebben dat er in de MAPS bijna altijd wel een nog 'heet' conferentieverslag opgenomen is.
Zoals reeds eerder gezegd, worden (indien mogelijk) alle TeX bijdragen in LaTeX omgezet. 'Auteursomgevingen' worden daarbij 'herschreven' in standaard LaTeX omgevingen, terwijl soms ook enkele speciale TeX commando's door een equivalente LaTeX vervangen worden. Daarbij blijkt dikwijls dat bij TeX bijdragen, de plain TeX code zodanig met de werkelijke tekst is verweven, dat het geheel nauwelijks meer herkenbaarheid heeft (komen we ook veel bij bepaalde PostScript files en programmerprogramma's tegen). Zeker in de huidige negentiger jaren moet zeker ook een TeXer gestructureerd en overzichtelijk kunnen werken.
Het uiteindelijke MAPS resultaat bestaat uit ongeveer 8 à 10 'include-files', met daarnaast een `maps.sty` waarin ondermeer de algemene typografische zaken zijn opgenomen. Dit alles moet dan enkele verwerkingsslagen ondergaan om zowel het geheel een wat redelijk uiterlijk te geven, de altijd nog aanwezige fouten er zo veel als mogelijk uit te halen, als ook om de ergste vormen van inconsistentheid te verwijderen.
Een aantal artikelen speciaal in deze MAPS gaven wederom wat lichte hoofdbrekens. Zo moest voor de bijdrage 'Introduction to MetaPost' de laatste versie van `dvips` worden geïnstalleerd (5.493 of hoger). Zo bevatte de uit WordPerfect afkomstige PostScript file (bijdrage van Huub Mulders) veel te lange records om goed via e-mail te kunnen verkrijgen. Ook de 'graphicsinput' macro van de SCIENTIFIC WORD bijdrage moest aangepast worden naar de in de MAPS stijl gebruikte `\psfig` en `\epsf` macro's (de `\special` macro wordt in de MAPS stijl niet gebruikt).
Tenslotte, na het verzamelen (en bijschaven) van enkele interessante 'netwerkfiles' is de klus dan weer geklaard. Het resultaat mag er met deze MAPS m.i. weer zijn. Wederom ondermeer een groot aantal min of meer algemene bijdragen, twee uitgebreide congresverslagen, veel PostScript, leuke macro's van David Salomon en een interessante verzameling werken van onze voorzitter t.b.v. de (bijna) TeX guru. Hem nu toch maar in een apart deel van de MAPS geplaatst; komt het beter tot zijn recht; hij verdient het.

6 Tot Slot

Het zal u nu duidelijk zijn dat voor een 'document' als de MAPS (de formele naam 'tijdschrift' wordt met opzet niet gebruikt), ondermeer de volgende taken aan de auteur zelf moet worden overgelaten:

- De auteur wordt aangeraden zijn bijdragen *bij voorkeur* in een logische documentstructuur aan te bieden (LaTeX, ltugboat.sty). Bij TeX documenten moet eventueel met (algemene) macro's gewerkt worden. De feitelijke tekst zelf moet zoveel mogelijk verschoond blijven van 'visuele' commando's. Het geheel behoort een overzichtelijk uiterlijk te hebben.
- De bijdragen dienen *bij voorkeur* zoveel mogelijk inhoudelijk correct te zijn (consistent taalgebruik; goede opbouw; etc). Het is daarom aan te raden om een bijdrage, voor de uiteindelijke inzending, altijd eerst door anderen te laten lezen.

Het is misschien nooit direct gezegd, doch het is min of meer logisch dat de verantwoordelijkheid van een bijdrage geheel bij de betreffende auteur ligt. Meningen van auteurs kunnen rechtlijnig staan t.o.v. die van de MAPS editor of t.o.v. het NTG bestuur, maar dat mag vanzelfsprekend niet een reden zijn een bijdrage geheel retour afzender te sturen. We leven nu eenmaal in een democratisch land.
Wij wensen u weer veel leesplezier met de MAPS die voor u ligt. Fouten (al dan niet typografisch) zult u ongetwijfeld tegen blijven komen. U kunt ze altijd doorsturen, doch bedenk daarbij ook dat zeker uw editor (bij lange na) niet volmaakt is.

[4] Jos Winnink staat altijd weer paraat en garant voor deze essentiële laatste verwerkingsslag.

Van de Voorzitter. . .

September 1992

1 TuG

Het is ronduit verheugend in 'The prez says. . . ' te lezen dat de stabiliteit van TEX etc. een groot goed is. Als dan ook nog de leiders van het lxiii project[1] hetzelfde in Praag, 'en plein public,' uitbazuinen, dan ben je blij. Even later vraag je je toch af, hoe het komt dat het zo lang geduurd heeft voordat deze lieden zich dit realiseerden. Was de verwarring van de kern met de verschillende user-interfaces misschien in het spel? Of speelde blinde toepassing van analogie-redenering—software die niet onderhouden wordt is gedoemd te sterven—een rol? TUGboat verschijnt weer regelmatig, en TTN rapporteert bijzonder fraai over het wel-en-wee van TEX etc, worldwide. De ukTEXug en NTG bieden een gecombineerd lidmaatschap met TUG.[2] TEXHaX wordt nu behartigd door Peter Abbott en collegas. De lxiii, driver, PD PC distributie projecten worden door TUG gesteund c.q. gesubsidieerd. Het organiseren van de annual meetings en het uitbrengen van de proceedings is in goede handen. De membership committee heeft de voordelen van het TUG lidmaatschap expliciet gemaakt en aangevuld. De T-shirts committee draait als nooit tevoren. Prima!

Desondanks gaat het nog niet goed met TUG. Na Nelson Beebe hebben wij Malcolm Clark als interim-prez.[3] Ron Whitney heeft per 1 September 1992 het office verlaten, en een leegte achter gelaten. Financieel moet er nog steeds bijgepast worden, gelukkig steeds minder. De BoD vergaderingen zijn nog verre van effectief. Het menselijk omgaan met actieve commissieleden, vooral als de aangedragen 'message' onwelkom is, heeft nog veel weg van het aloude de 'boodschap met de boodschapper' te verwarren, met al zijn kwalijke gevolgen. In Cork was ik getuige van (en dus gewaarschuwd, en in zekere mate ook medeplichtig door geen veto uit te spreken) de kwalijke manier waarop Ray Goucher 'bedankt werd voor bewezen diensten.' Het ledental is teruggelopen van ≈ 4.500 naar ≈ 3.500. De life-lijn van TUG—TUGboat—is organisatorisch-technisch zeer kwetsbaar, eigenlijk ontoelaatbaar kwetsbaar. De laatste boodschap is mij niet in dank afgenomen. Ook op klachten over de behandeling van auteurs-in-spe wordt niet adequaat gereageerd. Wie een artikel aanbiedt, heeft op zijn minst recht op informatie over de te doorlopen procedure en de daarbij behorende termijnen, zodat duidelijk wordt hoeveel tijd gaat verlopen tot acceptatie en publicatie. Ook de houding van sommige referees is voor verbetering vatbaar. Anders dan bij een wetenschappelijk tijdschrift gaat het bij een tijdschrift van een Users Group immers om wederzijdse hulp.[4] Het referee-en van boekbesprekingen gaat mij te ver. En laten wij wel wezen in het publishing gebeuren is TEX slechts een zijstroompje, hoog ontsprongen, en kristal helder, dat wel. Net zo als met schone lucht en zuiver bronwater is het vrij voor iedereen. Helaas wordt het ook relatief schaarser. De grote stroom van de dagelijkse publicaties bestaat uit de kranten en de tijdschriften, die zonder gebruik van TEX worden opgemaakt. Zelfs binnen het bastion van de wetenschap wordt de dienst uitgemaakt door Wordperfect, in ieder geval in Nederland. Oost- en Midden-Europa vormen hierop een uitzondering, wie weet voor hoe lang nog.

En verder, . . . de paradox is ontstaan dat de Amerikanen geen eigen LuG hebben.

2 Europese LuG-s

De Europese activiteit dit jaar was de EuroTEX'92, met CSTuG als spin in het Web.[5] Toch jammer dat CSTuG een tijdsbeklemming heeft aangezegd t.a.v. redistributie van het goede werk. Het prijsvoordeel van een grotere oplage van de proceedings rechtvaardigt niet de beperking van de vrijheid van redistributie van geselecteerd materiaal door andere LUG-s. Bovendien is de hoop op een grotere verkoop op deze manier een illusie. Ik schat in dat eerder het omgekeerde—tijdig geselecteerd materiaal *herdistribueren*—appetit tot aanschaf zal opwekken. Bovendien, er had toch even gepolsd kunnen worden hoeveel exemplaren NTG dacht te kunnen afnemen? Wij moeten kennelijk nog leren effectief met onze nieuwe media—lees netwerk—om te gaan, vooral dit soort eenvoudige polsingen zijn nu makkelijk mogelijk, waarmee financiële risicos gereduceerd kunnen worden.

[1] LATEX3 project: Frank Mittelbach and Chris Rowley.

[2] Er zijn 2-maal zoveel gecombineerde NTG-TuG leden als verwacht.

[3] Tot 1 Januari 1993. Voor de verkiezing zijn geen nominaties binnengekomen.

[4] Na uitdrukkelijke melding van het een en ander gaat het nu gelukkig al beter. Ik hoop echter dat de gedane suggesties standaard procedures zullen worden.

[5] Voor de proceedings ≈DM 30,–, neme men contact op met jvesely@cspguk11.bitnet. Het komt op de NTG leestafel ter inzage.

In Europa zijn DANTE en GUTenberg sterke stromingen. Zij hebben het mogelijk gemaakt dat afgevaardigden van Oost- en Midden-Europa konden deelnemen aan de Praagse EuroTEX. Een prima initiatief. Sommige Engelsen en ondergetekende voelden zich in verlegenheid gebracht toen dit in Praag bekend werd gemaakt. Waarom was dit niet eerder kenbaar gemaakt? Waarom werden wij niet van dit goede idee op de hoogte gebracht, zo dat ook wij konden helpen? Wij hebben er niet aan gedacht, helaas.[6] In ieder geval probeert Peter van Summeren de subsidies structureel via Brussel te laten verlopen, in het kader van de Europese integratie.

Naast de traditionele LuG-s zijn er nu ook een aantal Oost- en Midden-Europese groepen. CSTuG, Tsjechoslowakije, is goed georganiseerd met een redelijke basis van actieve leden. (Zij splitsen niet op!) GUST, Polen, met zo'n 100 leden, heeft veel potenties. CYRTuG heeft kennelijk een lange aanloop nodig. HUNTuG, Hongarije, blijft wat bescheiden. Roemenië komt net kijken, en danst vrolijk mee.

Volgend jaar is er *de* bijeenkomst in Aston, met helaas het geharrewar over de naam en door wie het georganiseerd gaat worden. De BoD-commissie heeft Aston als enige locatie voorgesteld, met als resultaat dat de annual TUG meeting volgend jaar in Europa is. Het is jammer dat DANTE dit niet volmondig steunt. Het zou toch zo mooi zijn: een gecombineerde TUG-EuroTEX bijeenkomst. Soms heeft het TEX-wereldje wat weg van een religieuze sekte, incluis al het gekibbel. Er is niets menselijks vreemd aan. Niettemin blijf ik van mening dat het net zo makkelijk plezieriger kan gaan.

3 NTG

Het grote gebeuren dit jaar was het contact met de wiskundigen: voorjaarsbijeenkomst bij het CWI, en stukjes in de mededelingen van het Wiskundig Genootschap. Misschien dat wij ons de komende tijd moeten richten op de contacten met SURF, met het 'PD TEX op de PC' als troef.[7] Als tweede grote activiteit was er de cursus van David Salomon: Insights and Hindsights. Ruim 45 NTG-ers hebben hun inzicht in de TEXnigma in een klap op een hoger plan gebracht. Prima deze investering in mensen, te meer daar wij een financiële investering van f 2000,– hadden begroot en er uiteindelijk positief met ruim f 1000,– zijn uitgesprongen. David is bezig zijn materiaal te herbewerken tot een boek in zijn 'sabbatical,' van September tot en met December. In het voorjaar brengen wij dan de tweede, herziene, versie uit, wederom als MAPS Special. Watch out!

Met GUST heb ik nader kennis gemaakt. Een informele samenwerking is afgetast. Informatie-uitwisseling, samenwerking en sharing van het goede werk en know-how, ça va sans dire. Ook gaan wij de bulletin's uitwisselen.[8] Er is echter een nadeel: wij kunnen zo slecht Pools lezen. Op de leestafel zal het GUST bulletin, to come, aardig staan, naast het Tsjechische bulletin, de vertrouwde Komödie, de GUTenberg cahiers, en hopelijk weer eens een Baskerville of TEX-line. Deze federatieve benadering stimuleert kleinschaligheid, met zijn creatieve spin-offs. Elke groep legt zijn accenten, met een rijke schakering als resultaat. Dit in tegenstelling tot de centralistische moloch, waarbij alles wat er even buiten valt wordt afgeknepen, en derhalve de dood is voor creativiteit. Prima toch, die federatieve gedachte!

Verder gaan wij bescheiden door, met als hoofdlijnen:

- onze voorjaars- en najaarsbijeenkomsten, met de bijbehorende MAPS-en;
- samenwerking (met zuster en wetenschappelijke verenigingen);
- de PD TEX etc. distributie voor PC-s;
- beschikbaar stellen van PR- en cursusmateriaal (MAPS specials);
- het behartigen van de file- en listserver(s);
- verbreding van de basis van actieve leden.

Het lustrum staat in het teken van de algemene typografie—Van font tot boek—en zal plaatsvinden in de regio Utrecht, op 10 juni 1993, met als gastinstelling het KNMI.

4 Literate Programming

George Greenwade heeft onlangs een discussielijst over literate programming geopend. Veel e-mail! Geïnteresseerden kunnen inhaken op

```
listserv@shsu.bitnet
```

middels de boodschap:

```
subscribe litprog "<Je naam>"
```

Het is mogelijk de geaccumuleerde discussie per maand op te vragen bij de fileserver in `shsu`.

Via deze lijst kwam ik in contact met Richard Kooijman, die kennelijk al wat ervaring heeft met Literate Programming. Als introductie tot Literate Programming is er Knuth's artikel uit 1984, en het boek van Wayne Sewell (1989): Weaving a program, Van Nostrand Reinhold. Knuth's artikel en '. . . a lot of other writings about styles and standards for reliable software engineering, is collected in the anthology

[6] Het is wel zo dat de GUTenberg organisatie vorig jaar diverse collegas subsidiëerde, en dus hadden wij dat wel kunnen extrapoleren.

[7] Dit ondanks dat de HIO-s e.d. nu ook bij SURF een mailbox kunnen huren, en daarmee toegang hebben tot het electronisch netwerk incluis het FTP-en.

[8] Dit ook met een wiskundig tijdschrift in Brno.

```
Knuth, D.E (1991): Literate Programming
CSLI Lecture Notes No 27
ISBN 0-9370-7380-6/4 (paper/cloth).'
```

Ten aanzien van de programmatuur het volgende van `Richard.Kooijman@dnpap.et.tudelft.nl`:

> 'Ik zelf gebruik NOWEB. Dit is een simpel LP tool die geschikt is voor alle programeertalen. Het kent slechts twee soorten blokken: docu en source. Docu gaat naar een TEX file met verbatim code. Code gaat naar een .c file (b.v. geef je aan in een Makefile).
>
> Voordeel is dat het geschikt is voor alle programmeertalen en het blijft van de source code formatering af. Zoals jij je code typt zo komt het eruit. Dit is tevens een nadeel, want de tool helpt je dus niet het formaat consistent te houden, b.v.
>
> ```
> if                if
> BEGIN                BEGIN
> ...       en      END
> END
> ```
>
> worden allebei gewoon gekopieerd naar de TEX docu en source code files.
>
> Andere tools helpen wel en kunnen zelfs gereserveerde keywords vet afdrukken om de source te verduidelijken. Nadeel daarvan is dat je je eigen instellingen moet kwijt kunnen in de tools (hoe moeten if's geformatteerd worden) en als je het voor BASIC wilt gebruiken (b.v.) dan moet er een BASIC definitie bestand aanwezig zijn.
>
> CWEB lijkt het meest op WEB, maar is alleen voor C sources.
>
> Spidery WEB lijkt ook veel op WEB en kent een aantal programmeertalen: C, C++, Ada, FORTRAN en Pascal (geloof ik).
>
> Nu over naar het belangrijkste punt: wat draait er op PC's? Alle systemen die ik hier genoemd heb draaien in ieder geval op UNIX, maar ik weet alleen van NOWEB zeker dat het op mijn PC draait. Ik heb het overgezet (C code was ANSI compatible, dus dat ging gelijk goed) maar er worden een aantal scriptjes gebruikt, zowel voor csh als awk.'

Mijns inziens is het grote vernieuwende van Literate Programming het *relationele* programmeren, in tegenstelling tot het vertrouwde sequentiële programmeren. Dat de link met de documentatie een belangrijke relatie is, en dat daardoor literate programming geïdentificeerd wordt met gelijktijdig documenteren, doet aan het algemene relationele niets af. Het is zo ongewoon, dat er nog heel wat ervaring mee moet worden opgedaan, om de voor- en nadelen boven water te krijgen.

Concept begroting van de Nederlandstalige TEX Gebruikersgroep

voor het jaar 1993

Hieronder vindt U de voorlopige begroting voor 1993 van de Nederlandstalige TEX gebruikersgroep. Een toelichting volgt na de tabel.

Inkomsten		**Uitgaven**	
Contributie	*f* 13.055,00	Administratie	*f* 600,00
Sponsoring		Kamer van Koophandel	*f* 61,00
Saldo NTG-dagen '93	PM	Bijeenkomsten	
Rente	*f* 445,00	Bestuurskosten	*f* 400,00
		Computerfaciliteiten	PM
		Nieuwsbrief/Verslagen	*f* 8.000,00
		Reis bijdragen	*f* 3.000,00
		Representatie	*f* 1.300,00
		Onvoorzien	*f* 139,00
	f 13.500,00		*f* 13.500,00

1 Toelichting

- **Inkomsten**:
  1. Contributie
     De post contributie is gebaseerd op het aantal leden in oktober 1992. Dat bedroeg:

30	instituten	30	× *f* 200,00	*f* 6.000,00
4	studenten	4	× *f* 50,00	*f* 200,00
50	personen	50	× *f* 75,00	*f* 3.750,00
46	personen	46	× *f* 67,50	*f* 3.105,00
				f 13.055,00

  2. Sponsoring
     Er wordt geen sponsoring verwacht.
  3. Saldo NTG-dagen
     In 1993 viert de NTG hoor vijfjarig bestaan met een lustrum bijeenkomst. Er is nog geen begroting gemaakt voor dit evenement. Het bestuur hoopt dat de inkomsten de uitgaven zullen dekken.
  4. Rente
     De vereniging heeft in de afgelopen jaren een behoorlijk kapitaal opgebouwd. Als dit niet nodig is om een tegenvaller op te vangen moet het mogelijk zijn behoorlijk wat rente te krijgen.

- **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 1991 en de realisatie in 1992 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* 20,00 per exemplaar.
  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. Als tegenprestatie wordt een verslag van de bijeenkomst verwacht, ter publicatie binnen de vereniging.
  7. Representatie
     Als bestuursleden van zustervereningingen bij onze bijeenkomst uitgenodigd worden wordt een tegemoetkoming in de kosten gegeven. Het bestuur is van plan in 1993 de 'PR sets' te produceren. De kosten hiervan zullen naar schatting *f* 3,50 per exemplaar bedragen. Uitgaande van een oplage van 300 stuks betekent dat een uitgave van *f* 1.050,–.
  8. Onvoorzien
     Spreekt voor zich.

Naast hierboven vermelde direkte uitgaven is het bestuur van plan een deel van het kapitaal van de NTG te gebruiken om te investeren in een voorraad van het cursusmateriaal van David Salomon en in het opzetten van een distributie van een PD versie van TEX (and friends) voor PCs (in de ruime zin van het woord). Het is de bedoeling dat dit soort zaken budget-neutraal zijn.

NTG's listserver TEX-NL

13 oktober 1992

TeX-NL is de Nederlandstalige TEX-informatie distributielijst (ook wel discussielijst genoemd). Het adres is:
`TEX-NL@NIC.SURFNET.NL`

Men kan zich op deze TeX-NL discussielijst abonneren (TEX-NL mails ontvangen en versturen) door het versturen van de volgende één-regelige e-mail:

```
to      : listserv@nic.surfnet.nl
subject :'any'
SUBSCRIBE TEX-NL your_name
```

Een lijst van deelnemers is te verkrijgen door het versturen van de volgende één-regelige e-mail:

```
to      : listserv@nic.surfnet.nl
subject : 'any'
REVIEW TEX-NL
```

Met als resultaat:

```
*
*  TEX-NL
*
*  Review=       Public
*  Subscription= Open
*  Send=         Public
*  Notify=       Yes
*  Reply-to=     List,Ignore
*  Files=        Yes
*  Validate=     Store only
*  Errors-To=    Owners
*  X-Tags=       Comment
*  Stats=        None,Private
*  Confidential= No
*
*  owner=  Quiet:,U070007@HNYKUN11  (Niek Cox)
*  owner=  Quiet:,BRAAMS@HLSDNL5    (Johannes Braams)
*  owner=  EVERS@HUTRUU53           (Evert Jan Evers)
pfuetz@AGD.FHG.DE                        Matthias Pfuetzner, ZGDV Darmstadt
VDBERG@ALF.LET.UVA.NL                    Martin H. vdBERG
KROPVELD@AMC.UVA.NL                      Dani"el Kropveld
CI@ANALYSIS.RUG.AC.BE                    Chris Impens
LMO@AUTOCTRL.RUG.AC.BE                   Benedict R. Verhegghe
raichle@AZU.INFORMATIK.UNI-STUTTGART.DE  Bernd Raichle
LAAAA18@BLEKUL11                         Erik van Eynde
GRAD205@BRFUEM                           Students of Mathematics at BRFUEM
HJBORTOL@BRLNCC                          Humberto Jose Bortolossi
C11000@BRUSPVM                           CRISTIANO CORDARO
FDC@CAGE.RUG.AC.BE                       "F. De Clerck"
bakker@CS.RULIMBURG.NL                   Harm Bakker
piet@CS.RUU.NL                           Piet van Oostrum
eijkhout@CS.UTK.EDU                      Victor Eijkhout
vansoest@CS.UTWENTE.NL                   Dick C. "van Soest"
A.G.Geraets@CTRL.PHYS.TUE.NL             Tonnie Geraets
frankw@CWI.NL                            Frank van de Wiel
rvdh@CWI.NL                              Rob van der Horst
TNNDCVZ@CYCL.PHYS.TUE.NL                 C. van Zwijnsvoorde
ernst@DCMR1.UUCP                         E.R. de Vreede
Leendert.Combee@DELFT.GECO.SLB.COM       leendert combee
combee@DELGEO.NL                         leendert combee
X33@DHDURZ1                              Joachim Lammarsch
andreas@DUTENTB.ET.TUDELFT.NL            Andreas A. Buykx
nust@DUTENTB.ET.TUDELFT.NL               Jan H Nusteling
abi@DUTIAA.TUDELFT.NL                    Ton Biegstraaten
wim@DUTIOSA.TUDELFT.NL                   Wim Penninx
witajgb@DUTISTA.TUDELFT.NL               Hans Braker
wiorst5@DUTIWS.TUDELFT.NL                Bert van Zomeren
dvb@DUTMPW1.TUDELFT.NL                   Diederik van Batenburg
mkmfhuy@DUTREX.TUDELFT.NL                Tom Huijgen
mkmfipa@DUTREX.TUDELFT.NL                Edgar Iparraguirre
wbtrvos@DUTREX.TUDELFT.NL                Ron v. Ostayen
kees@DUTTWTA.TUDELFT.NL                  C.L. Koster
robk@DUTTWTA.TUDELFT.NL                  Rob Kuyper
andries@DUTW6.TUDELFT.NL                 jans andries
```

gerard@DUTW9.TUDELFT.NL	Gerard Kuiken
jaap@DUTW9.TUDELFT.NL	Jaap van der Zanden
martien@DUTW9.TUDELFT.NL	Martien Hulsen
eikelboom@ECN.NL	Jaap Eikelboom
hogenbirk@ECN.NL	Alfred Hogenbirk
vanderstad@ECN.NL	Rob C. L. van der Stad.
vannes@ECN.NL	Gerard van Nes
winnink@ECN.NL	J.J. Winnink
DOLLY@ECO.RUG.NL	Wietse Dol
FRAMBACH@ECO.RUG.NL	"Erik Frambach"
MALLY@ECO.RUG.NL	Maarten H. van der Vlerk
N.POPPELIER@ELSEVIER.NL	"Nico Poppelier"
alex@ET.KULEUVEN.AC.BE	Alex Schoenmakers
ludo@ET.KULEUVEN.AC.BE	Vangilbergen Ludo
AJKRIJGSMAN@ET.TUDELFT.NL	Ardjan Krijgsman
COMBEE@ET.TUDELFT.NL	leendert combee
rafel@FENK.WAU.NL	Rafel Israels
huygen@FGG.EUR.NL	Paul E.M. Huygen
vdende@FGG.EUR.NL	Jan van der Ende
BOLDY@F2.NHL.NL	Mike Boldy
INEKE_VAN@GCRC2.WUSTL.EDU	ineke vandermeulen
roosmalen@GLAS06.DECNET.PHILIPS.NL	Martin van Roosmalen
A3530004@HASARA11	Hans Verhey.
A401INEK@HASARA11	ineke weijer
A410SAKE@HASARA11	Sake J. Hogeveen
A471BERN@HASARA11	Bernard R. Bollegraaf
A471HANS@HASARA11	hans van der meer
A9530020@HASARA11	repke de vries
SOND0016@HASARA11	R Veldhuyzen van Zanten
EMMEN@HASARA5	Ad Emmen
DENHAAN@HDETUD5	Jack den Haan
RCRONH@HEITUE5	Ron Helwig
ELEICZ@HEITUE51	C. van Zwijnsvoorde
ALDHAHIR@HENUT5	Alaaddin Al-Dhahir
CGL@HGRRUG5	CG VAN DER LAAN
DRUNEN@HGRRUG5	Rudi van Drunen
KONING@HGRRUG5	RUUD H. KONING
BOSVELD@HGRRUG51	"Gerard Bosveld"
STOOP@HGRRUG51	"Paul Stoop"
KANABY@HHEOUH51	Abdy Jooya
APPRMB@HHEOUH53	Rut Berns
LETTVA@HLERUL2	Andrea de Leeuw van Weenen
FTHKOPER@HLERUL52	GER KOPER
BORSBOOM@HLERUL53	G.J.J.M. Borsboom
VDSCHOOT@HLERUL53	Jan Vanderschoot
DAVID@HLERUL59	David van Leeuwen
U001290@HNYKUN11	Niek Cox
U001310@HNYKUN11	Ronald Kappert
U070040@HNYKUN11	Patrick Wever
U212307@HNYKUN11	Peter Bronts
U212757@HNYKUN11	Mathieu Koppen
U216002@HNYKUN11	Paul Wackers
U250005@HNYKUN11	Peter-Arno Coppen
U251006@HNYKUN11	Hans Stoks
U253002@HNYKUN11	Constant Cuypers
U439019@HNYKUN11	Ton de Haan
U605005@HNYKUN11	Willem Jan Karman
U605008@HNYKUN11	Rik Fleuren
U641012@HNYKUN11	Rini van Doorn
BISON@HNYKUN52	PIETER BISON
CAOS@HNYKUN52	HENS BORKENT
SYLVIA@HNYMPI51	"Sylvia Aal"
GPTEX@HRZ.UNI-GIESSEN.DBP.DE	TeX-Inst., HRZ Univ. Giessen, F.R.G.
Guenter.Partosch@HRZ.UNI-GIESSEN.DBP.DE	Guenter Partosch, HRZ Univ. Giessen, F
SURF083@HTIKUB5	Johannes de Moor
S172HMUL@HTIKUB5	Huub Mulders
EVERS@HUTRUU53	Evert Jan Evers / Rijksuniv. Utrecht
KETTENIS@HWALHW5	"Di(r)k Kettenis"
VDVELDEN@HWALHW5	Mark van der Velden
erikjan@ICCE.RUG.NL	Erik-Jan Vens
stokhof@ILLC.UVA.NL	Martin Stokhof
ITALIANO@IMEUNIV	Antonio ITALIANO
E.H.M.Ulijn@IO.TUDELFT.NL	Erik H.M. Ulijn
HAAN@IRIVAX.TUDELFT.NL	Henk de Haan
devries@KNMI.NL	Hans de Vries
BORDEWIJK@KVI.NL	JOHAN BORDEWIJK
POL@KVI.NL	"John van Pol"
STAPEL@KVI.NL	"Kees Stapel"
AnneMarie.Mineur@LET.RUU.NL	Anne-Marie Mineur
Jules.vanWeerden@LET.RUU.NL	Jules van Weerden, RUU
WILLEMSE@LETT.KUN.NL	Rijk Willemse
wierda@LTB.BSO.NL	Gerben Wierda
andre@MAESTRO.HTSA.AHA.NL	Andre v.d. Vlies
NSEV@MARIN.NL	<E.F.G. van Daalen>
R.H.M.Huijsmans@MARIN.NL	rene huijsmans
bnb@MATH.AMS.COM	Barbara Beeton
M.DRUIF@MATH.RULIMBURG.NL	Marlies Haenen
demeijer@MATH.RUU.NL	Andre J. de Meijer
hvdberg@MATH.UTWENTE.NL	Harmen van den Berg
soos@MATH.UTWENTE.NL	Adwin Soos
twpolder@MATH.UTWENTE.NL	Jan Willem Polderman
bob@MPI.KUN.NL	Bob Boelhouwer
hdavids@MSWE.DNET.MS.PHILIPS.NL	Henk Davids
MBR@OCE.NL	Marius Broeren
mourad@PH.TN.TUDELFT.NL	Mourad Bouchahda
mwijz@PHYS.UVA.NL	Maurits Wijzenbeek
SZAJOW@PLWRTU11	Krzysztof Szajowski
d5@PMS.UIA.AC.BE	Benoit Suykerbuyk
vdkoijk@RADTH.RUU.NL	John van der Koijk
MOUCHE@RCL.WAU.NL	Pierre von Mouche
WEITS@RCL.WAU.NL	Ello Weits

```
nibr!mark@RELAY.NLUUG.NL                  Mark van Veen
J.L.Braams@RESEARCH.PTT.NL                Johannes L. Braams
egdnt01hbtex@RUG.NL                       Henk Brouwer
DINGS@RUGR86.RUG.NL                       Marcel Dings
nijhof@RUGTH4.TH.RUG.NL                   Jeroen Nijhof
rein@RUG4.CS.RUG.NL                       Rein Smedinga
VDGRIEND@RULWINW.LEIDENUNIV.NL            J.A. van de Griend
ELBERS@SARA.NL                            Chris Elbers
lenssen@SG.TN.TUDELFT.NL                  K.-M. Lenssen
v912182@SI.HHS.NL                         Eric Veldhuyzen
POPPE@SWOV.NL                             "Frank Poppe"
HOEB@TUDW02.TUDELFT.NL                    J.B.W. HOEBEEK
HUISMAN@TUDW03.TUDELFT.NL                 H. Huisman
eleipb@URC.TUE.NL                         Phons Bloemen
rcpt@URC.TUE.NL                           Piet Tutelaers
robin@UTAFLL.UTA.EDU                      R. Cover
KALPAKLI@UWAV1.U.WASHINGTON.EDU           Mehmet Kalpakli   Kalpakli@uwav1.u.was
Wilfred=Zegwaard%Alg%AAE.WAU@VINES.WAU.NL  Wilfred Zegwaard
KNAPPEN@VKPMZD.KPH.UNI-MAINZ.DE           Joerg Knappen Uni-Mainz
KNAPPEN@VKPMZD.PHYSIK.UNI-MAINZ.DE        J"ORG KNAPPEN
SPIT@VM.CI.UV.ES                          Werenfried Spit
dee@WLDELFT.NL                            Dick Dee
Marc.Kool@WLDELFT.NL                      Marc H. Kool
vdvoorn@WLDELFT.NL                        Marjan v\d Vooren
best@ZEUS.RIJNH.NL                        Robert W. Best
*
* Total number of "concealed" subscribers:         2
* Total number of users subscribed to the list:  162  (non-"concealed" only)
* Total number of local node users on the list:    0  (non-"concealed" only)
*
```

Opmerkingen:

- Verzocht wordt om de TEX-NL listserver niet te gebruiken voor het versturen van grote bestanden (programma's) indien van het alternatief: **de TEX-NL fileserver** (*zie bijlage G*), gebruik gemaakt kan worden.
- Daar ook enkele buitenlanders meeluisteren, wordt men verzocht de 'subject' van de mail in het Engels op te geven.
- De TEX-NL listserver is bij uitstek geschikt voor ondermeer een ondersteuningsverzoek bij een TEX/LATEX/driver probleem, voor vragen over beschikbaarheid van bepaalde software modulen, voor aankondigingen van bijeenkomsten en/of cursussen, voor het attenderen op bepaalde publicaties, voor het attenderen op bepaalde produkten en voor een mededeling die ook voor een grotere groep interessant is.
- Daar het versturen van e-mail's zowel voor het netwerk als voor de ontvanger een duidelijke belasting is, wordt men verzocht geen 'overbodige' boodschappen te versturen zoals bijvoorbeeld 'bedankt', 'geheel mee eens' en dergelijk.
- Ondanks het feit dat het opnemen van een vraag bij de beantwoording dikwijls verhelderend kan werken, dient de verhouding 'antwoord' tot 'vraag' binnen de redelijke proporties te blijven.
- Indien problemen optreden bij het opzeggen dan wel het wijzigen van het eigen e-mail adres op de listserver, wordt men verzocht contact op te nemen met de beheerder E.J. Evers.

NTG's fileserver TEX-NL

13 oktober 1992

Sinds mei 1989 heeft NTG de TEX-NL fileserver. Voor leden interessante files worden daarbij centraal beschikbaar gesteld.
Men kan files van deze fileserver betrekken door het sturen van een volgende e-mail:

```
to       : listserv@nic.surfnet.nl
subject  : 'any'
GET filename1 filetype1
GET filename2 filetype2
GET filename3 filetype3
etc
```

Waarbij de mogelijke *filenames* en *filetypes* in de hieronder getoonde listing zijn opgenomen.

De lijst van alle aanwezige files is te verkrijgen door het sturen van de volgende e-mail:

```
to       : listserv@nic.surfnet.nl
subject  : 'any'
GET TEX-NL FILELIST
```

Door het versturen van bovenstaande één-regelige boodschap ontvangt men de volgende informatie terug:

```
*  TEX-NL FILELIST for LISTSERV@HEARN.
*  TeX-NL Filelist
*
*  Contains
*      -- general TeX stuff (implementations for micros, graphical
*         shells, printer drivers, etc.)
*      -- specifically Dutch stuff (styles and options, hyphenation
*         patterns)
*      -- Dutch TeX Users Group (NTG) stuff
*
* Please Note:
*         To prevent having to send large files across the networks,
*         the uuencoded zoo archives will be split if they are larger
*         then 1024 records. In these cases the command
*         GET <name> PACKAGE will send all the parts to the requestor.
*
*  ***************************************************************
*
*  This file lists the programs that are stored on LISTSERV and can be
*  retrieved by network users.
*
*  If an entry shows nrecs=0 the file is not available.
*
*  This filelist may be sorted in columns 47 to 63 to get a list of
*  files in the order of their updates. Sorting in descending order
*  shows the most recently updated files at the top.
*
*
*  NTG= U641000@HNYKUN11 Victor Eijkhout
*  NTG= U641001@HNYKUN11 Victor Eijkhout
* ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
*
*  The GET/PUT authorization codes shown with each file entry describe
*  who is authorized to GET or PUT the file:
*
*     ALL = Everybody
*     N/A = Not Applicable
*     LCL = Local users, as defined at installation time
*     PRV = Private, ie list members
*     OWN = List owners
*     NAD = Node Administrators, ie official BITNET/EARN contacts
*     CTL = LISTEARN Controllers (Also called "Postmasters")
*
*:    NTG = 'BRAAMS@HLSDNL5',   /*  Johannes Braams                  */
*:          'BRAAMS@HLSDNL50',  /* Johannes Braams                   */
*:          'BRAAMS@HLSDNL51',  /* Johannes Braams                   */
*:          'BRAAMS@HLSDNL52',  /* Johannes Braams                   */
*:          'EVERS@HUTRUU53'    /* Evert Jan Evers                   */
*
* ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

**************************************************************************
*
*  Dutch hyphenation patterns
*
*  Hyphen1 TeX : shortened Celex-list, all lines with a 5 in them removed,
```

```
*               in order to be able to load it when you can't stretch
*               the 'triesize'
*  Hyphen2 TeX : long and powerful (author: Celex, Nijmegen)
*               Note that this requires stretching the 'triesize'
*               of both TeX and IniTeX!
*  Hyphen3 TeX : The (very short) patterns for Dutch created by Peter Vanroose
*  UShyphen ADD: extra patterns to handle the Tugboat exception log
*               (author: Gerard Kuiken)
*
**************************************************************************
*                              rec               last - change
* filename filetype   GET PUT -fm lrecl nrecs   date     time   File description
* -------- --------   --- --- --- ----- ----- -------- -------- ----------------
  HYPHEN1  TEX        ALL NTG V      80  6122 91/05/03 20:00:23
  HYPHEN2  TEX        ALL NTG V      80  7945 91/05/04 10:07:40
  HYPHEN3  TEX        ALL NTG V      80   338 91/05/03 19:56:55
  USHYPHEN ADD        ALL NTG V      73   378 90/05/14 13:20:11

**************************************************************************
*
*  Options for Dutch
*
*  A4 STY    : A4-paper width and height
*              by Nico Poppelier and Johannes Braams (historical order)
*              Note that this is not the A4 option of John Pavel.
*  A4 TeX and A4 DOC: Accompanying documentation for A4.STY
*  Dutch old : Redefines captions and does other useful things for
*              all standard document styles. (author: Johannes Braams)
*              This is really an international option.
*              This file has been superseded by the dutch.sty in the
*              BABEL system (See further on)
*  German STY: The style on which 'Dutch' was based. The two are
*              compatible. (author: Hubert Partl) version 2.3e
*  Sober STY : Reduces section headings and white spaces a bit;
*              this is only repair for the standard styles. The official
*              NTG styles (below) can do without. (author: Nico Poppelier)

**************************************************************************
*                              rec               last - change
* filename filetype   GET PUT -fm lrecl nrecs   date     time   File description
* -------- --------   --- --- --- ----- ----- -------- -------- ----------------
  A4       STY        ALL NTG V      80   135 91/02/13 10:50:05
  A4       DOC        ALL NTG V      80   511 91/02/13 13:58:32
  A4       TEX        ALL NTG V      80    57 92/08/26 11:09:48
  DUTCH    OLD        ALL NTG V      80   397 90/12/20 18:45:23
  GERMAN   STY        ALL NTG V      80   627 91/11/06 11:17:50
  SOBER    STY        ALL NTG V      77   147 89/06/24 16:06:16

**************************************************************************
*
*  The BABEL system
*
*      This is the BABEL system as it is described in TUGboat.
*
*      See the file BABEL README for further instructions
*      The file BABEL BUG lists bugreports and comments since 8/7/91
*      The files BABEL UA?ZOO contain all files.
*   (they can be ordered by sending "GET BABEL PACKAGE" to LISTSERV)
*
**************************************************************************
  BABEL    README     ALL NTG V      80   128 92/01/20 18:33:56
  BABEL    $PACKAGE   ALL NTG V      80     5 91/11/05 09:22:42
  BABEL    UAAZOO     ALL NTG V      80  1024 91/11/05 09:27:10
  BABEL    UABZOO     ALL NTG V      80  1024 91/11/05 09:28:36
  BABEL    UACZOO     ALL NTG V      80  1024 91/11/05 09:30:34
  BABEL    UADZOO     ALL NTG V      80  1024 91/11/05 09:32:41
  BABEL    UAEZOO     ALL NTG V      80   693 91/11/05 09:33:47
  BABEL    BUG        ALL NTG V      80   120 91/08/21 23:36:33
  BABEL    TEX        ALL NTG V      80    52 92/02/20 10:30:37
  BABEL    DOC        ALL NTG V      80   755 91/08/21 14:55:02
  BABEL    COM        ALL NTG V      80   229 91/08/21 14:55:16
  HYPHEN   DOC        ALL NTG V      80   325 91/08/21 23:08:08
  BABEL    HYPHEN     ALL NTG V      80   105 91/08/21 23:06:54
  BABEL    SWITCH     ALL NTG V      80    80 91/08/21 23:07:07
  BABEL22  SWITCH     ALL NTG V      80    88 91/08/21 23:07:21
  BABEL32  SWITCH     ALL NTG V      80    87 91/08/21 23:07:56
  LANGUAGE DAT        ALL NTG V      80     6 91/05/22 01:52:28
  LATEXHAX DOC        ALL NTG V      80   102 91/08/21 14:55:29
  LATEXHAX COM        ALL NTG V      80    58 91/08/21 14:55:41
  ESPERANT DOC        ALL NTG V      80   213 91/08/21 14:58:02
  ESPERANT STY        ALL NTG V      80    99 91/08/21 14:58:22
  DUTCH    DOC        ALL NTG V      80   517 91/08/21 14:58:51
  DUTCH    STY        ALL NTG V      80   158 91/08/21 14:59:13
  ENGLISH  DOC        ALL NTG V      80   260 91/08/21 15:00:33
  ENGLISH  STY        ALL NTG V      80   115 91/08/21 15:01:30
  GERMANB  DOC        ALL NTG V      80   707 91/08/21 15:01:59
  GERMANB  STY        ALL NTG V      80   259 91/08/21 15:02:55
  FRANCAIS DOC        ALL NTG V      80   599 92/02/17 16:40:47
  FRANCAIS STY        ALL NTG V      80   262 91/09/21 23:05:37
  ITALIAN  DOC        ALL NTG V      80   211 92/02/17 16:26:15
  ITALIAN  STY        ALL NTG V      80    99 91/08/21 15:03:57
  PORTUGES DOC        ALL NTG V      80   236 91/08/21 15:04:16
  PORTUGES STY        ALL NTG V      80   108 91/08/21 15:04:31
  SPANISH  DOC        ALL NTG V      80   751 92/01/25 17:15:03
  SPANISH  STY        ALL NTG V      80   209 92/01/25 17:13:58
  DANISH   DOC        ALL NTG V      80   211 91/08/21 15:05:29
  DANISH   STY        ALL NTG V      80    99 91/08/21 15:05:49
  NORSK    DOC        ALL NTG V      80   256 91/08/21 15:06:02
  NORSK    STY        ALL NTG V      80   121 91/08/21 15:06:19
  SWEDISH  DOC        ALL NTG V      80   216 91/08/21 15:06:35
  SWEDISH  STY        ALL NTG V      80    99 91/08/21 15:06:54
```

```
  FINNISH  DOC        ALL NTG V      80   214 91/08/21 15:07:17
  FINNISH  STY        ALL NTG V      80   100 91/08/21 15:07:36
  MAGYAR   DOC        ALL NTG V      80   232 91/08/21 15:08:00
  MAGYAR   STY        ALL NTG V      80   108 91/08/21 15:08:13
  CROATIAN DOC        ALL NTG V      80   198 92/02/17 16:42:01
  CROATIAN STY        ALL NTG V      80   100 91/08/21 15:08:48
  CZECH    DOC        ALL NTG V      80   235 91/08/21 15:09:10
  CZECH    STY        ALL NTG V      80   108 91/08/21 15:09:24
  POLISH   DOC        ALL NTG .       .     0 ........ ........
  POLISH   STY        ALL NTG .       .     0 ........ ........
  ROMANIAN DOC        ALL NTG V      80   211 91/08/21 15:09:38
  ROMANIAN STY        ALL NTG V      80    99 91/08/21 15:10:00
  SLOVENE  DOC        ALL NTG V      80   214 91/08/21 15:10:14
  SLOVENE  STY        ALL NTG V      80    99 91/08/21 15:10:30
  RUSSIAN  DOC        ALL NTG V      80   443 91/08/21 15:10:42
  RUSSIAN  STY        ALL NTG V      80   166 91/08/21 15:11:04
  CYRILLIC DOC        ALL NTG V      80   298 91/08/21 15:11:45
  CYRILLIC STY        ALL NTG V      80   137 91/08/21 15:12:07
*************************************************************************
*
*  Dutch styles (author: Victor Eijkhout)
*
*  Completely compatible to 'article' and 'report', but improved layout;
*  these styles have as default language English,
*  for Dutch or German add corresponding style options
*
*  Artikel1 doc : Article-compatible, tight look, documented (somewhat)
*  Artikel1 sty : without documentation
*  Artikel2 doc : Article-compatible, heavily indented; quite something else
*  Artikel2 sty : without documentation
*  Artikel3 doc : Article-compatible; zero parindent, positive parskip;
*                 otherwise similar to Artikel1
*  Artikel3 sty : without documentation
*  Rapport1 doc : Report-compatible; looks like Artikel1
*  Rapport1 sty : without documentation
*  Rapport2 doc : will probably not come into being.
*  Rapport2 sty : without documentation
*  Rapport3 doc : Report-compatible; looks like Artikel3
*  Rapport3 sty : without documentation
*  Boek doc     : Book-compatible; artikel1 layout
*  Boek sty     : without documentation
*
*  Options for the Dutch styles
*
*  Ntg10 doc    : 10point option for all styles
*  Ntg10 sty    : without documentation
*  Ntg11 doc    : 11point option for all styles
*  Ntg11 sty    : without documentation
*  Ntg12 doc    : 12point option for all styles
*  Ntg12 sty    : without documentation
*  Voorwerk doc : Replaces Titlepage.STY for report styles
*  Voorwerk sty : without documentation
*
*  NTGstyle UA? : All in one buy; UUencoded ZOO archive (see below
*                 for ZOO)
*   (they can be ordered by sending "GET NTGSTYLE PACKAGE" to LISTSERV)
*************************************************************************
*                              rec               last - change
* filename filetype   GET PUT -fm lrecl nrecs   date     time   File description
* -------- --------   --- --- --- ----- ----- -------- -------- ----------------
  ARTIKEL1 DOC        ALL NTG V      80  1344 92/08/25 23:15:51
  ARTIKEL1 STY        ALL NTG V      80   712 92/08/25 23:20:17
  ARTIKEL2 DOC        ALL NTG V      80  1304 92/09/04 13:34:41
  ARTIKEL2 STY        ALL NTG V      80   675 92/09/04 13:35:52
  ARTIKEL3 DOC        ALL NTG V      80  1379 92/08/25 23:17:52
  ARTIKEL3 STY        ALL NTG V      80   722 92/08/25 23:18:40
  RAPPORT1 DOC        ALL NTG V      80  1668 92/08/25 23:25:12
  RAPPORT1 STY        ALL NTG V      80   854 92/08/25 23:21:09
  RAPPORT2 DOC        ALL NTG .       .     0 ........ ........
  RAPPORT3 DOC        ALL NTG V      80  1666 92/08/25 23:23:38
  RAPPORT3 STY        ALL NTG V      80   846 92/08/25 23:22:08
  BOEK     DOC        ALL NTG .       .     0 ........ ........
  BOEK     STY        ALL NTG V      80   682 91/02/21 11:24:43
  NTG10    DOC        ALL NTG V      80   193 92/01/15 23:22:17
  NTG10    STY        ALL NTG V      80   166 92/01/15 23:23:14
  NTG11    DOC        ALL NTG V      80   197 92/01/15 23:22:44
  NTG11    STY        ALL NTG V      80   169 92/01/15 23:23:28
  NTG12    DOC        ALL NTG V      80   196 92/01/15 23:22:58
  NTG12    STY        ALL NTG V      80   170 92/01/15 23:23:42
  VOORWERK DOC        ALL NTG .       .     0 ........ ........
  VOORWERK STY        ALL NTG V      80    78 92/02/07 00:07:44
  NTGSTYLE $PACKAGE   ALL NTG V      80     7 92/01/16 01:01:09
  NTGSTYLE UAA        ALL NTG V      80  1000 92/08/26 11:19:20
  NTGSTYLE UAB        ALL NTG V      80  1000 92/08/26 11:19:49
  NTGSTYLE UAC        ALL NTG V      80  1000 92/08/26 11:21:16
  NTGSTYLE UAD        ALL NTG V      80  1000 92/08/26 11:22:47
  NTGSTYLE UAE        ALL NTG V      80  1000 92/08/26 11:23:52
  NTGSTYLE UAF        ALL NTG V      80  1000 92/08/26 11:28:38
  NTGSTYLE UAG        ALL NTG V      80   709 92/08/26 11:29:22

*************************************************************************
*
*  The letter style according to Dutch NEN norms (by Victor Eijkhout)
*
*  BRIEF STY    : The style file
*  BRIEF TeX    : An example letter
*  BRIEFDOC TeX : Explanation of the options of the letter style
*
*************************************************************************
*                              rec               last - change
* filename filetype   GET PUT -fm lrecl nrecs   date     time   File description
```

```
* -------- --------   --- --- --- ----- ----- -------- -------- ----------------
  BRIEF    STY        ALL NTG V      80   709 92/03/31 21:07:15
  BRIEF    TEX        ALL NTG V      80   199 92/03/31 20:54:46
  BRIEFDOC TEX        ALL NTG V      80   294 92/03/31 20:55:06

*************************************************************************
*
*  The latest in TeXnology
*
*  ASCII TeX    : ASCII table (author: Victor Eijkhout)
*
*  BTXMAC.TEX   : BibTeX 0.99c macros for use with plain TeX.
*                 The file specifies that is meant for TeX 3.0 or later
*
*  DUTCH BST    : BibTeX style v 1.11 for Dutch by Werenfried Spit
*  DUTCH2 BST   : BibTeX style v 2.0 for Dutch by Werenfried Spit
*                 this needs the harvard files
*
*  HARVARD README : short description of what's in harvard.zoo and where
*                   it came from.
*  HARVARD UUE    : A uuencoded zoo archive containing 6 files
*
*  CHNGEBARS    : Michael Fine's changebar.sty, modified for use with plain
*                 TeX as well as with LaTeX. Also modified to support DVItoPS
*                 \specials as well as DVI2LN3 \specials
*
*************************************************************************
*                               rec               last - change
* filename filetype   GET PUT -fm lrecl nrecs   date     time   File description
* -------- --------   --- --- --- ----- ----- -------- -------- ----------------
  ASCII    TEX        ALL NTG V      80   190 91/06/26 22:43:05
  BTXMAC   TEX        ALL NTG V      80   624 90/08/15 16:59:21
  DUTCH    BST        ALL NTG V      80  1413 91/11/14 14:19:43
  DUTCH2   BST        ALL NTG V      80  1459 92/03/17 22:53:35
  HARVARD  README     ALL NTG V      80    44 92/03/13 19:08:30
  HARVARD  UUE        ALL NTG V      80   730 92/03/13 19:10:28
*-----------------------------------------------------------------------------
*
*  TUGBOAT CMN  : Common commands for Tugboat styles
*  TUGBOAT STY  : Plain TeX style for Tugboat article
*  LTUGBOAT STY : LaTeX style for Tugboat articles
*  TUBGUIDE TEX : A guide for auhtors
*
*  TUGPROC STY  : Plain TeX style file for the proceedings of TuG meetings
*  LTUGPROC STY : LaTeX TeX style file for the proceedings of TuG meetings
*                 Both files need the Tugboat files
*  GUIDEPRO TEX : A guide for authors
*
*-----------------------------------------------------------------------------
  TUG      $PACKAGE   ALL NTG V      80     7 92/03/11 23:40:39
  TUGBOAT  CMN        ALL NTG V      80   894 92/03/11 23:11:19
  TUGBOAT  STY        ALL NTG V      80  2351 92/03/11 23:13:35
  LTUGBOAT STY        ALL NTG V      80   600 92/03/11 23:14:19
  TUBGUIDE TEX        ALL NTG V      80   844 92/03/11 23:38:18
  TUGPROC  STY        ALL NTG V      80   357 92/03/11 23:14:39
  LTUGPROC STY        ALL NTG V      80   193 92/03/11 23:14:50
  GUIDEPRO TEX        ALL NTG V      80   933 92/03/11 23:39:28
*-----------------------------------------------------------------------------
* CHANGEBAR     : Changebar style file for LaTeX 2.09
*                 Changebar V3.0
*                 Supports DVItoLN03, DVIps, DVItoPS, DVIdrv (v1.5+)
*                 Documentation uses doc.sty
*-----------------------------------------------------------------------------
  CHANGBAR $PACKAGE   ALL NTG V      80     4 92/01/15 01:38:21
  CHANGBAR BUG        ALL NTG V      80    79 92/01/16 00:03:27
  CHANGBAR DRV        ALL NTG V      80    76 92/03/13 12:41:47
  CHANGBAR DOC        ALL NTG V      80  1216 92/01/15 01:40:00
  CHANGBAR STY        ALL NTG V      80   333 92/01/15 01:40:22
*                 Changebar V2.? to be removed soon
  CHNGBARS STY        ALL NTG V      80   881 91/06/16 16:02:05
*-----------------------------------------------------------------------------
*  LATEX PACKAGE : Latest versions of all LaTeX materials;
*                  UUencoded ZOO archive
*                  Release 1 december 1991
*-----------------------------------------------------------------------------
  LATEX    $PACKAGE   ALL NTG V      80    11 91/12/02 16:15:01
  LATEX    UAA        ALL NTG V      80  1010 92/04/21 13:48:03
  LATEX    UAB        ALL NTG V      80  1010 92/04/21 13:49:54
  LATEX    UAC        ALL NTG V      80  1010 92/04/21 13:53:02
  LATEX    UAD        ALL NTG V      80  1010 92/04/21 13:58:57
  LATEX    UAE        ALL NTG V      80  1010 92/04/21 14:05:09
  LATEX    UAF        ALL NTG V      80  1010 92/04/21 14:13:05
  LATEX    UAG        ALL NTG V      80  1010 92/04/21 14:15:16
  LATEX    UAH        ALL NTG V      80  1010 92/04/21 14:23:01
  LATEX    UAI        ALL NTG V      80  1010 92/04/21 14:27:33
  LATEX    UAJ        ALL NTG V      80  1010 92/04/21 14:31:03
  LATEX    UAK        ALL NTG V      80  1010 92/04/21 14:35:28
  LATEX    UAL        ALL NTG V      80  1010 92/04/21 14:43:38
  LATEX    UAM        ALL NTG V      80   947 92/04/21 14:55:01
*-----------------------------------------------------------------------------
*  LATEXFON PACKAGE: Latest versions of all LaTeX fonts;
*                    UUencoded ZOO archive
*                    Release 1 december 1991
*-----------------------------------------------------------------------------
  LATEXFON $PACKAGE   ALL NTG V      80     2 91/12/02 16:18:52
  LATEXFON UAA        ALL NTG V      80   640 92/02/05 12:56:35
  LATEXFON UAB        ALL NTG V      80   386 92/02/05 12:57:35
*-----------------------------------------------------------------------------
*  NFSS PACKAGE : The New Font Selection Scheme as published by
*                 Frank Mittelbach and Rainer Schoepf
*                 Release 1 december 1991
```

```
*------------------------------------------------------------------------------
  NFSS     $PACKAGE   ALL NTG V      80     4 91/12/02 16:19:52
  NFSS     UAA        ALL NTG V      80  1000 91/12/02 15:18:44
  NFSS     UAB        ALL NTG V      80  1000 91/12/02 15:20:31
  NFSS     UAC        ALL NTG V      80  1000 91/12/02 15:23:21
  NFSS     UAD        ALL NTG V      80    83 91/12/02 15:22:50
*------------------------------------------------------------------------------
*  MULTICOL     : The multicolumn package written by Frank Mittelbach and
*                 Rainer Schoepf, as published in TUGboat.
*                 The packacge includes DOC.STY. The package consists of three
*                 files, MULTICOL README, MULTICOL UAAZOO, MULTICOL UABZOO.
*                 These files must be distributed together.
*   (they can be ordered by sending "GET MULTICOL PACKAGE" to LISTSERV)
*
*------------------------------------------------------------------------------
  MULTICOL $PACKAGE   ALL NTG V      80     3 92/01/06 17:10:25
  MULTICOL README     ALL NTG V      80    82 91/11/06 11:18:05
  MULTICOL UAAZOO     ALL NTG V      80  1000 91/11/06 11:19:06
  MULTICOL UABZOO     ALL NTG V      80   775 91/11/06 11:21:13
*------------------------------------------------------------------------------
*  SUPERTAB     : Theo Jurriens' supertabular.sty for creating tables longer
*                 than one page. Modified by Gabriele Kruljac and Johannes
*                 Braams. Now also supports different tablehead on first page
*                 and different tabletail on last page of the table.
*                 Note: supertabular.doc is *NOT* meant for FMi's doc option
*------------------------------------------------------------------------------
  SUPERTAB DOC        ALL NTG V      80   506 92/07/10 10:24:13
  SUPERTAB STY        ALL NTG V      80   285 92/07/10 10:24:41
  SUPERTAB TEX        ALL NTG V      80   234 91/04/25 17:37:27
*------------------------------------------------------------------------------
*  CMRULE       : "The TeX Ruler" by Victor Eykhout using cm-fonts
*  PSRULE       : "The TeX Ruler" by Victor Eykhout using PostScript fonts
*                 Both files contain uuencoded dvi-files
*------------------------------------------------------------------------------
  CMRULE   UUE        ALL NTG V      80  1008 91/07/15 17:17:01
  PSRULE   UUE        ALL NTG V      80  1019 91/07/15 18:07:46
*------------------------------------------------------------------------------
*  NASSFLOW UUE : A uuencoded ZOO archive containing style options for
*                 nassi-schneidermann diagrams or flow-diagrams.
*                 Man-pages are included in the archive.
*                 The file NASSFLOW README lists what is available.
*------------------------------------------------------------------------------
  NASSFLOW README     ALL NTG V      80    37 91/06/21 14:37:16
  NASSFLOW UUE        ALL NTG V      80   600 91/07/05 10:22:33

**********************************************************************
*
*    'Fun with TeX'
*
*    The files below were collected at the NTG meeting in Eindhoven,
*    in november 1991. The meeting was devoted to 'Fun with TeX'
*    Hanna Ko{\l}odziejska presented her GO macros and fonts.
*    Daniel Taupin spoke about MusicTeX.
*    Both packages are provided here.
*
**********************************************************************
*------------------------------------------------------------------------------
*    The MusicTeX package is stored as multi-part UUencoded ZOO archives
*    It contains the macros, the METAFONT files and the fonts.
*
*    The following files are provided:
*
*    MusicTeX README with a description of what is in the ZOO files and some
*                    comments on how to install everything
*
*    MusicTeX UA?    contains TeX and Metafont sources as well as examples
*    MusicPK  UA?    contains PK files
*    Recueil  UA?    contains a dvi file that can be printed when the fonts
*                    are installed.
*
*------------------------------------------------------------------------------
*                              rec               last - change
* filename filetype   GET PUT -fm lrecl nrecs   date     time   File description
* -------- --------   --- --- --- ----- ----- -------- -------- ----------------
  MUSICTEX $PACKAGE   ALL NTG V      80    27 92/01/21 16:11:34
  MUSICTEX README     ALL NTG V      80    65 92/01/16 00:02:21
  MUSICTEX UAA        ALL NTG V      80  1000 92/01/15 23:30:30
  MUSICTEX UAB        ALL NTG V      80  1000 92/01/15 23:31:57
  MUSICTEX UAC        ALL NTG V      80  1000 92/01/15 23:33:54
  MUSICTEX UAD        ALL NTG V      80  1000 92/01/15 23:35:15
  MUSICTEX UAE        ALL NTG V      80  1000 92/01/15 23:36:40
  MUSICTEX UAF        ALL NTG V      80  1000 92/01/15 23:38:06
  MUSICTEX UAG        ALL NTG V      80  1000 92/01/15 23:39:33
  MUSICTEX UAH        ALL NTG V      80  1000 92/01/15 23:41:03
  MUSICTEX UAI        ALL NTG V      80    61 92/01/15 23:40:47
  MUSICPK  UAA        ALL NTG V      80  1000 92/01/15 23:43:04
  MUSICPK  UAB        ALL NTG V      80  1000 92/01/15 23:44:11
  MUSICPK  UAC        ALL NTG V      80  1000 92/01/15 23:45:37
  MUSICPK  UAD        ALL NTG V      80  1000 92/01/15 23:47:08
  MUSICPK  UAE        ALL NTG V      80  1000 92/01/15 23:48:39
  MUSICPK  UAF        ALL NTG V      80  1000 92/01/15 23:50:00
  MUSICPK  UAG        ALL NTG V      80   824 92/01/15 23:51:13
  RECUEIL  UAA        ALL NTG V      80  1000 92/01/15 23:53:03
  RECUEIL  UAB        ALL NTG V      80  1000 92/01/15 23:54:12
  RECUEIL  UAC        ALL NTG V      80  1000 92/01/15 23:55:36
  RECUEIL  UAD        ALL NTG V      80  1000 92/01/15 23:57:00
  RECUEIL  UAE        ALL NTG V      80  1000 92/01/15 23:58:39
  RECUEIL  UAF        ALL NTG V      80  1000 92/01/16 00:01:37
  RECUEIL  UAG        ALL NTG V      80  1000 92/01/16 00:03:10
  RECUEIL  UAH        ALL NTG V      80    11 92/01/16 00:02:07
*------------------------------------------------------------------------------
```

```
*    The GO package is stored as a two-part UUencoded ZOO archive
*    It contains the macros, the METAFONT files and the source for
*    the article as it appeared in the MAPS 91.2.
*-------------------------------------------------------------------------------
  GO       $PACKAGE   ALL NTG V      80     3 91/11/28 09:54:19
  GO       README     ALL NTG V      80    35 91/11/28 09:54:56
  GO       UAA        ALL NTG V      80   768 91/11/28 10:03:36
  GO       UAB        ALL NTG V      80   349 91/11/28 10:04:05

*******************************************************************************
*
*  Pleasant reading material about TeX and its uses
*
*  NTGstyle TeX : Manual for the Dutch LaTeX styles
*  Layout TeX   : Article about documentstyle development in LaTeX
*                 Intended as supplement to chapter 5 LaTeX book
*  Layout2 TeX  : Goes with previous; in German (Hubert Partl)
*  Refman STY   : Needed for previous two
*  Bridge TeX   : About setting bridge games in LaTeX (Kees van der Laan)
*  Artdoc TeX   : The history of the 'Artikel' styles; almost a
*                 manual for document style development; in Dutch
*  Rapdoc TeX   : The same for the 'Rapport' styles (Victor Eijkhout)
*  Gentle TeX   : A Gentle Introduction to TeX (Michael Doob)
*  Gentrev TeX  : A review of "A Gentle..." by C.G. van der Laan with
*                 comments by Michael Doob
*
*  TTN00    TEX : Het eerste nummer van 'TeX and TUG News'' a prototype issue'
*  TTV1N1   TEX : Het eerste echte nummer van 'TeX and TUG News''
*  TUGNEWS  STY : De bijbehorende style file
*
*******************************************************************************
*                              rec               last - change
* filename filetype   GET PUT -fm lrecl nrecs   date     time   File description
* -------- --------   --- --- --- ----- ----- -------- -------- ----------------
  NTGSTYLE TEX        ALL NTG V      72   122 89/09/04 12:20:21
  LAYOUT   TEX        ALL NTG V      79  1090 89/06/26 12:12:34
  LAYOUT2  TEX        ALL NTG V      80  1011 89/06/26 12:03:15
  REFMAN   STY        ALL NTG V      79   492 90/03/26 18:12:17
  BRIDGE   TEX        ALL NTG V      75   415 89/06/26 11:54:36
  ARTDOC   TEX        ALL NTG V      71   708 89/09/04 12:21:36
  RAPDOC   TEX        ALL NTG V      75   735 89/09/04 12:22:13
  GENTLE   TEX        ALL NTG V      79  5341 90/01/09 13:56:01
  GENTREV  TEX        ALL NTG V      80   380 91/11/05 09:39:07
  TTN00    TEX        ALL NTG V      80  2047 91/06/04 16:54:07
  TTNV1N1  TEX        ALL NTG V      80  1969 92/02/26 00:46:24
  TUGNEWS  STY        ALL NTG V      80    92 92/02/26 00:43:10

*******************************************************************************
*
*  Nederlandstalige TeX Gebruikersgroep (Dutch TeX Users Group)
*
*  Notuul1 TeX  : Vergadering 23 juni 1988
*  Notuul2 TeX  : Vergadering 24 november 1988
*  Notuul3 TeX  : Vergadering 11 mei 1989 (3 bestanden: notuul3a,b,c)
*  TeXdag89 TeX : Verslag eerste Nederlandse TeXdagen 29/30 juni 1989
*
*  Statuten TeX : De statuten van de vereniging NTG
*  Statuten sty : bijbehorende document stijl-optie
*
*******************************************************************************
*                              rec               last - change
* filename filetype   GET PUT -fm lrecl nrecs   date     time   File description
* -------- --------   --- --- --- ----- ----- -------- -------- ----------------
  NOTUUL1  TEX        ALL NTG V      80  1043 89/06/26 11:50:54
  NOTUUL2  TEX        ALL NTG V      80  1457 89/06/26 11:52:06
  NOTUUL3A TEX        ALL NTG V      80   108 89/10/30 10:12:47
  NOTUUL3B TEX        ALL NTG V      80  1941 89/10/30 10:13:27
  NOTUUL3C TEX        ALL NTG V      80  2634 89/10/30 10:14:05
  TEXDAG89 TEX        ALL NTG V      73   274 89/12/08 13:04:52
  STATUTEN TEX        ALL NTG V      80   532 91/03/04 20:55:17
  STATUTEN STY        ALL NTG V      80    94 91/03/04 14:14:21

*******************************************************************************
*
*  TeX for micros
*
*  STZOO UUE    : UUencoded ARC archive with ZOO for the Atari ST
*  MSZOO UA?    : zoo.exe version 2.1 for MS-DOS, UUencoded
*  MSFIZ UUE    : fiz.exe version 2.1 for MS-DOS, UUencoded
*  ZOO2-1 UA?   : Sources for zoo version 2.1. A uuencoded zoo archive
*  WP2LATEX UUE : WordPerfect to LaTeX translator, ZOOed
*
*******************************************************************************
*                              rec               last - change
* filename filetype   GET PUT -fm lrecl nrecs   date     time   File description
* -------- --------   --- --- --- ----- ----- -------- -------- ----------------
  STZOO    UUE        ALL NTG F      61  2181 89/12/14 12:33:31
  MSZOO    UAA        ALL NTG V      80   900 92/07/09 13:10:31
  MSZOO    UAB        ALL NTG V      80   657 92/07/09 13:12:19
  MSFIZ    UUE        ALL NTG V      80   412 92/06/22 09:12:32
  ZOO2-1   UAA        ALL NTG V      80   900 92/06/22 09:17:14
  ZOO2-1   UAB        ALL NTG V      80   900 92/06/22 09:19:06
  ZOO2-1   UAC        ALL NTG V      80   900 92/06/22 09:21:13
  ZOO2-1   UAD        ALL NTG V      80   900 92/06/22 09:23:20
  ZOO2-1   UAE        ALL NTG V      80   900 92/06/22 09:24:55
  ZOO2-1   UAF        ALL NTG V      80   900 92/06/22 09:26:50
  ZOO2-1   UAG        ALL NTG V      80   900 92/06/22 09:28:42
  ZOO2-1   UAH        ALL NTG V      80   102 92/06/22 09:28:11
  WP2LATEX UUE        ALL NTG V      80  1229 90/03/12 15:58:03

*******************************************************************************
```

```
*
*    Utility programs
*
*    UUE    .C    : An (almost) foolproof uuencode program that protects
*                   against network character conversions. It can also
*                   split a large file in multiple parts
*                   This program is used in creating the uue-files in
*                   in this filelist.
*    UUD    .C    : An (almost) foolproof uudecode program that can
*                   correct chaacter conversions. It automagically glues
*                   the parts created by uue.c together.
*    UUX    .DOC  : (Some) documentation to the above programs.
*
**************************************************************************
  UUE      C          ALL NTG V      80   298 92/01/24 11:30:21
  UUD      C          ALL NTG V      80   732 91/12/24 22:25:34
  UUX      DOC        ALL NTG V      80   134 91/12/24 22:25:50

*-------------------------------------------------------------------------
*    AMSPEL20 UA% : A Uuencoded *zip* archive that contains a
*                   spelling checker for pc's
*-------------------------------------------------------------------------
  AMSPEL20 UAA        ALL NTG V      80   900 92/08/10 12:48:10
  AMSPEL20 UAB        ALL NTG V      80   900 92/08/10 12:49:35
  AMSPEL20 UAC        ALL NTG V      80   900 92/08/10 12:51:29
  AMSPEL20 UAD        ALL NTG V      80   900 92/08/10 12:52:58
  AMSPEL20 UAE        ALL NTG V      80   420 92/08/10 12:53:31

**************************************************************************
*
*    METAFONT sources
*
*    AMSREAD.ME   : A few notes about the contents of AMSFONTS.UUE
*    AMSFONTS.UUE : The AMS font collection UUencoded ZOO archive
*                   split in ten pieces of app. 100kByte
*
*    NOTE : This is still version 2.0 of the distribution !
*
**************************************************************************
*                              rec               last - change
* filename filetype   GET PUT -fm lrecl nrecs   date     time   File description
* -------- --------   --- --- --- ----- ----- -------- -------- ----------------
  AMSREAD  ME         ALL NTG V      74    45 90/08/02 14:18:33
  AMSFONTS UU1        ALL NTG F      80  1644 90/08/02 15:50:09
  AMSFONTS UU2        ALL NTG F      80  1643 90/08/02 15:58:11
  AMSFONTS UU3        ALL NTG F      80  1643 90/08/02 16:07:46
  AMSFONTS UU4        ALL NTG F      80  1643 90/08/02 16:41:15
  AMSFONTS UU5        ALL NTG F      80  1643 90/08/02 16:44:37
  AMSFONTS UU6        ALL NTG F      80  1643 90/08/02 16:47:16
  AMSFONTS UU7        ALL NTG F      80  1643 90/08/02 16:49:31
  AMSFONTS UU8        ALL NTG F      80  1643 90/08/02 16:51:56
  AMSFONTS UU9        ALL NTG F      80  1643 90/08/02 16:53:58
  AMSFONTS UUA        ALL NTG F      80  1659 90/08/02 16:55:44
```

Behalve via de fileserver TEX-NL, zijn files ook te verkrijgen bij ondermeer de volgende ftp centra:

- `archive.cs.ruu.nl`
- `ftp.th-darmstadt.de`
- `labrea.stanford.edu`
- `math.utah.edu`
- `rusinfo.rus.uni-stuttgart.de`
- `tex.ac.uk`

of via e-mail bij:

- `mail-server@cs.ruu.nl`
- `mail-server@rusmv1.rus.uni-stuttgart.de`
- `LISTSERV@DHDURZ1`
- `texserver@tex.ac.uk`
- `tuglib@math.utah.edu`

Voor NTG leden die niet op een netwerk zijn aangesloten, kunnen de meeste files via diskettes verkregen worden. Nadere informatie hierover bij Gerard van Nes.

Werkgroep 3: Evaluatie
Formules in WP5.1, DECwrite en LaTeX

Huub Mulders
DRC – KUB

20 augustus 1990

1 Inleiding

De pakketten WP5.1, DECwrite en LaTeX bieden de mogelijkheid om formules te *typesetten*. Om de mogelijkheden en kwaliteit te kunnen beoordelen is geprobeerd een vijftal formules met behulp van de drie pakketten te maken.
Het invoeren van formules is in principe overal hetzelfde. De formule wordt, voorzien van *markups*, als tekst ingevoerd, bijvoorbeeld:

$$\sum_{n=1}^{\infty}$$

kan in WP worden gemaakt door de volgende tekst in te typen (de markups staan in hoofdletters):

```
SUM FROM {n=1} TO INF
```

Het gebruik van *markups* wijkt sterk af van de *normale* werkwijze in WP en DECwrite waarin met funktietoetsen en/of menu-keuzes gewerkt wordt.
Daar in LaTeX alle TeX *markups* voor formules gebruikt mogen worden en er in het werken met LaTeX en TeX ook geen verschil is kan onderstaand overal waar LaTeX staat ook TeX worden gelezen.

2 Formule-editors

- WP
  Bij WP is een speciale *formule-editor* voor het invoeren van formules, die vanuit WP met behulp van een reeks functie-toetsen wordt aangeroepen.
  De 'edit' mogelijkheden zijn zeer beperkt. In de WP *formule-editor* kan alleen gebruik gemaakt worden van de *functie* voor speciale karakters ([Ctrl 2]), maar bestaat niet de mogelijkheid om bijvoorbeeld een groter of kleiner lettertype te kiezen, of een stukje tekst te dupliceren.
  Bij de WP *formule-editor* horen een aantal tabellen waarin de *markups* staan en waaruit men een keuze kan maken.
  Voor het selekteren van een *markup* is in WP vaak een veelvoud van toetsaanslagen nodig vergeleken bij het intypen van de markup, daar staat tegenover dat men ondersteunt wordt met tabellen op het scherm. Bijvoorbeeld voor het selekteren van de *markup* **sum** moet het volgende gedaan worden:
  toets F5(lijst), daarna (afhankelijk van welke tabel op het scherm verschijnt) nul, een of meerdere keren de toets PgDn of PgUp aanslaan en tenslotte met de pijltjes toets naar het goede 'vakje' gaan waarna nog een Return gegeven moet worden.
  De *markups* maken een integraal onderdeel uit van het WP-bestand en kunnen dus alleen in WP aangemaakt en veranderd worden.
- DECwrite
  Bij DECwrite is een speciale *formule-editor* voor het invoeren van formules, die vanuit DECwrite (menu-keuze, 'link equation') moet worden aangeroepen.
  De 'edit' mogelijkheden zijn zeer beperkt, er is niet veel meer aanwezig dan: `select, cut, copy, paste`. Het zou beter zijn als de editor welke in DECwrite 'zelf' gebruikt wordt ook voor het maken van formules gebruikt zou kunnen worden.
  In de *formule-editor* van DECwrite kan gebruik gemaakt worden, via keuze menu, van een drietal tabellen waarin in een groot aantal (lang niet alle) *markups* staan.
  Daar in DECwrite formules (om aanpassingen te kunnen maken) in ASCII-files (in principe elke formule in een aparte file) moeten staan, kan natuurlijk ook van editors buiten DECwrite om gebruik gemaakt worden. Men mist dan wel de previewfaciliteiten.
  Wat geïmplementeerd is lijkt sterk op TeX, maar is een zeer beperkte subset, waarbij de *markups* soms ook nog niet precies hetzelfde effect hebben als bij TeX.
- **LaTeX**
  LaTeX-files bevatten uitsluitend plain ASCII-karakters en kunnen dan ook met behulp van gewone editors, op allerlei systemen, worden aangemaakt.

3 Previewers

- WP
  In WP kan bij het maken van een formule op elk moment, met behulp van een *previewer*, bekeken worden hoe de formule er uit ziet, de *previewer* wordt door een funktie-toets binnen de *formule-editor* opgeroepen. Er is nogal een verschil, in *typesetting* van de formule, tussen wat de previewer laat zien en wat uiteindelijk op de printer komt. Hoe het er op de printer uitziet is weer afhankelijk van de soort (type,merk) printer.
- DECwrite
  In DECwrite kan bij het maken van een formule op elk moment bekeken worden hoe de formule er uit komt te zien. De *previewer* wordt in de *formule-editor* via een keuze-menu opgeroepen. De verschijningsvorm op het scherm komt nagenoeg overeen met die op de printer.
  Een DECwrite document kan alleen naar een *PostScript*-printer.
- LaTeX
  Bij LaTeX moet een appart programma 'gedraaid' worden (dus buiten de editor) om te kunnen zien hoe de formule er uit ziet. Zoals de formule op het scherm komt, zo zal de formule ook op de printer komen (afgezien van afrondingsfouten i.v.m. het verschil van het aantal dot/inch van scherm t.o.v printer).

We geven op de volgende bladzijden de resultaten van een vijftal formules. Voor elk pakket wordt ook de bijbehorende *markup*-tekst gegeven. Dit deel van dit artikel kon alleen met LaTeX, in kombinatie met PostScript, gemaakt worden.

4 Opmerkingen bij de formules

4.1 Opmerkingen bij WP

- Geen mogelijkheid om de karaktergrootte te veranderen, alleen automatisch voor machten e.d. (*formule 3*, $\stackrel{\text{def}}{=}$)
- Alleen onder- en bovenstrepen kan over meer dan één karakter, maar b.v. hoed(hat), slingertje(tilde) of liggende accolades(brace) (*formule 1*) over meer dan een karakter ontbreken
- Er is maar één maat voor de pijlen (te korte pijltjes bij *formule 4*).
- Er wordt nogal wat kreativiteit verwacht. In de formules zijn de volgende *trukjes* gebruikt:
  - *formule 1*
    Extra hakenpaar ({..}) *om* **underline**.
    from of **to** door een spatie (~) laten volgen.
  - *formule 4*
    Een index bij een spatie (b.v. ~**_rho**).
    Verlengen van een horizontaal pijltje door er een streepje voor te zetten (– **HORZ** – **20** →)
    Veel gebruik gemaakt van **horz**, **vert** en **phantom** om onderdelen van de formule op de 'goede' plaats te krijgen
- Omdat er in WP de markup **MATRIX** is , gaf *formule 4* weinig problemen. Elk element in de matrix moet minstens één karakter bevatten, vandaar dat er nogal wat spaties (~) gebruikt moeten worden.
- Het afdrukken neemt erg veel tijd in beslag. Voor de vijf formules naar PostScript was ongeveer een half uur nodig en voor naar een HP-laserjet ruim tien minuten.

4.2 Opmerkingen bij DECwrite

- Het positioneren van de formules gaf vele problemen.
  Voor *formule 0* is als oplossing gekozen om over de formule spaties te zetten, en voor de andere *formules* om er blanko regels overheen te typen. Hierdoor werden de formules bij tekst wijzigingen verminkt. De formules in *frames* zetten was ook een mogelijke oplossing geweest, maar dan ontstonden er andere problemen.
  Het op hoogte positioneren van formules ten op zichte van de omringende tekst (b.v. in *textmode* de basis lijn gelijk) moet handmatig, wat niet echt goed te doen is.
- Geen mogelijkheid voor n^{de} machtswortel, b.v. $\sqrt[5]{x+5}$ (*formule 1*)
- a. geen markup voor maken van vertikale witruimte.
  b. voor horizontale witruimte slechts drie markups die een *vaste* witruimte geven
- Een zeer belangrijke *markup* die nodig is voor het maken van een matrix (tabel) ontbreekt (in LaTeX `\begin{array}...\end{array}`). Dit is een van de reden waarom *formule 4* niet afgemaakt is
- Alleen onder- en bovenstrepen kan over meer dan één karakter, maar b.v. hoed(hat), slingertje(tilde) of liggende accolades(brace) (*formule 1*) over meer dan een karakter ontbreken.
- Geen handige manier gevonden voor het wijzigen van formules.
  De gebruikte methode: gooi de oude formule weg in het document en *link* de formule opnieuw (*link equation*) en dan weer opnieuw positioneren.
- De *typesetting* laat veel te wensen over.
  Er zitten nog bugs in, zoals: griekse letters vet geven verkeerde karakters (`\bf\Lambda` geeft **£**, *formule 0*), vertikale dubbele streep in *formule 4* wordt in tweeën gesplitst.

4.3 Opmerkingen bij LaTeX

- Bij *formule 2* is gebruik gemaakt van de TeX-macro **\mathstrut** om de liggende akkolades (overbraces) op gelijke hoogte te krijgen.
- Bij *formule 3* kon $\Big\}_k$ alleen met TeX *markups* gemaakt worden.
- Geen problemen met *formule 4*.

5 De formules

5.1 De formules in WP

Formule 0

De mathematische expressie $S^{-1}TS-dg(\omega_1,\ldots,\omega_n)=\Lambda$ in *textmode*.

Formule 1

$$\sum_{i=1}^{\infty}\frac{1}{2^i}=1,\quad \{\overbrace{a,\ldots,a}^{k\ a's},\overbrace{b,\ldots,b}^{l\ b's}\},\quad \sqrt[3]{1+\sqrt{1+\sqrt{1+x}}},$$

Formule 2

$$A=\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix},\quad \left(\int_{-\infty}^{\infty} e^{-x^2}dx\right)^2=\pi,$$

Formule 3

$$2\uparrow\uparrow k \underset{=}{def} 2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}\Big\}k,\quad \frac{f(x+\Delta x)-f(x)}{\Delta x}\to f'(x)\ \text{as}\ \Delta x\to 0.$$

Formule 4

$$\begin{array}{ccccccccc} & & & & & & 0 & & \\ & & & & & & \downarrow & & \\ 0 & \longrightarrow & O_c & \xrightarrow{\iota} & E & \xrightarrow{\rho} & L & \longrightarrow & 0 \\ & & \| & & \downarrow\phi & & \downarrow\psi & & \\ 0 & \longrightarrow & O_C & \longrightarrow & \pi_* O_D & \xrightarrow{\delta} & R^1 f_* O_V(-D) & \longrightarrow & 0 \\ & & & & & & \downarrow\ \Theta_i\otimes\gamma^{-1} & & \\ & & & & & & R^1 f_*\left(O_V(-iM)\right)\otimes\gamma^{-1} & & \\ & & & & & & \downarrow & & \\ & & & & & & 0 & & \end{array}$$

5.2 De formules in DECwrite

Formule 0

De matematische expressie $\mathbf{S}^{1}\mathbf{TS} = \mathbf{dg}(\omega_1, \ldots, \omega_n) = \mathbf{\pounds}$ in *textmode*.

Formule 1

$$\sum_{i=1}^{\infty} \frac{1}{2^i} = 1, \{\underbrace{\overbrace{a, \ldots, a}^{k\ a's}, \overbrace{b, \ldots, b}^{l\ b's}}_{k+l\ elements}\}, \sqrt{1 + \sqrt{1 + \sqrt{1 + x}}}$$

Formule 2

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}, \quad \left(\int_{-\infty}^{\infty} e^{-x^2}\, dx \right)^2 = \pi,$$

Formule 3

$$2 \uparrow\uparrow k \stackrel{\text{def}}{=} 2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}} \Big\} k, \frac{f(x+\Delta x) - f(x)}{\Delta x} \rightarrow f'(x) \text{ as } \Delta x \rightarrow 0.$$

Formule 4

$$\begin{array}{ccccc} 0 & \longrightarrow & \mathcal{O}_c & \xrightarrow{\iota} & \\ & & \| & & \\ 0 & \longrightarrow & \mathcal{O}_C & \rightarrow & \end{array}$$

("gestopt, te moeilijk")

5.3 De formules in LaTeX

Formule 0

De mathematische expressie $\mathbf{S}^{-1}\mathbf{TS} = \mathbf{dg}(\omega_1, \ldots, \omega_n) =$ in *textmode.*

Formule 1

$$\sum_{i=1}^{\infty} \frac{1}{2^i} = 1,\ \{\overbrace{a, \ldots, a}^{k\ a's}, \overbrace{b, \ldots, b}^{l\ b's}\}_{k+l \text{ elements}},\ \sqrt[3]{1 + \sqrt{1 + \sqrt{1 + x}}},$$

Formule 2

$$A = \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \ldots & a_{mn} \end{pmatrix},\ \left(\int_{-\infty}^{\infty} e^{-x^2}\, dx \right)^2 = \pi,$$

Formule 3

$$2 \uparrow\uparrow k \stackrel{\text{def}}{=} 2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}} \Big\}k,\ \frac{f(x + \Delta x) - f(x)}{\Delta x} \to f'(x) \text{ as } \Delta x \to 0..$$

Formule 4

$$\begin{array}{ccccccccc} & & & & & & 0 & & \\ & & & & & & \downarrow & & \\ 0 & \longrightarrow & \mathcal{O}_c & \xrightarrow{\iota} & \mathcal{E} & \xrightarrow{\rho} & \mathcal{L} & \longrightarrow & 0 \\ & & \| & & \downarrow{\scriptstyle \phi} & & \downarrow{\scriptstyle \psi} & & \\ 0 & \longrightarrow & \mathcal{O}_C & \longrightarrow & \pi_*\mathcal{O}_D & \xrightarrow{\delta} & R^1 f_* \mathcal{O}_V(-D) & \longrightarrow & 0 \\ & & & & & & \downarrow{\scriptstyle \theta_i \otimes \gamma^{-1}} & & \\ & & & & & & R^1 f_* \left(\mathcal{O}_V(-iM)\right) \otimes \gamma^{-1} & & \\ & & & & & & \downarrow & & \\ & & & & & & 0 & & \end{array}$$

6 De tekst van de formules

6.1 De WP formules

Formule 0

```
{BOLD S}^1{BOLD TS}={BOLD dg}(omega_1,DOTSLOW,omega_n)= BOLD LAMBDA
```

Formule 1

```
sum from {i=1} to inf 1 over {2^i}=1,HORZ 80
\{{underline{ {overline{a,dotslow,a}}from~to {k~a's},
              {overline{b,dotslow,b}}from~to {l~b's}
            }
  }from{k+l~element's}to~
\},HORZ 80 nroot{3}{1+sqrt{1+sqrt{1+x}}},
```

Formule 2

```
A=left( matrix{ a_{11}& DOTSLOW &a_{1n}        #
                DOTSVERT & DOTSDIAG & DOTSVERT #
                a_{m1} & DOTSLOW & a_{mn}
              }
  right),horz 80
 left( INT_{Îinf}^inf e^{Îx^2} dx right)^2 = pi,
```

Formule 3

```
2 ↑ ↑ k HORZ 15 STACK {def#=}HORZ 15 LEFT.2^{2^{2^{.^{.^{.^2}}}}}
   RIGHT\}  VERT 55 k VERT -55, HORZ 80
{f(x+ DELTA x) -f(x)} OVER {DELTA x} → f'(x)~FUNC as~ DELTA x → 0.
```

Formule 4

```
MATRIX{~&~&~&~&~&0                                              #
       ~&~&~&~&~& ↓                                             #
       0& – HORZ-20→^&  O_c & VERT 50
            STACK{~_iota#– HORZ-20→ }
            &E& VERT 50 STACK{~_ rho#– HORZ-20→} & L &
            – HORZ-20→ &0                                       #
       ~&~& PARALLEL &~&PHANTOM{phi}↓ phi&~&PHANTOM{psi}↓ psi    #
       0& – HORZ-20→ & O_C& – HORZ-20→
            &pi_*O_D&VERT 50 STACK{~_ delta# – HORZ-20→}
            &R^1f_*O_V(-D)& – HORZ-20→& 0                      #
       ~&~&~&~&~&~&PHANTOM{\ _{THETA_i}\ _OTIMES \ _{gamma^{-1}}}↓
            VERT 50 \ _{THETA_i}\ _OTIMES\ _{gamma^{-1}}        #
       ~&~&~&~&~&~&R^1f_*LEFT(O_V(-iM)RIGHT)OTIMES gamma^{-1}   #
       ~&~&~&~&~&~&↓                                            #
       ~&~&~&~&~&~&0
     }
```

6.2 De DECwrite formules

Formule 0

```
${\bf S^{\rm-1}TS=dg}(\omega_1,\ldots,\omega_n)=\bf\Lambda$
```

Formule 1

```
\sum _{i=1}^\infty {1\over{2^i}}=1,\;
\{ \vcenter{\underline{ {{k\ a's}\atop\overline{\displaystyle a,\ldots,a}},
                        {{l\ b's}\atop\overline{\displaystyle b,\ldots,b}}
                      }\atop{\scriptstyle k+l\ elements}}\},\;
\sqrt{1+\sqrt{1+\sqrt{1+x}}},
```

Formule 2

```
A=\left({a_{11}\atop{\vdots\atop {\displaystyle a_{m1}}}}
        {\ldots\atop{ \ddots\atop\ldots}}
        {a_{1n}\atop{\vdots\atop{\displaystyle  a_{mn}}}}
  \right),
\ \left(\int_{-\infty}^\infty e^{-x^2}\,dx\right)^2=\pi,
```

Formule 3

```
2|uparr|uparr k {{\rm def}\atop=}
2^{2^{2^{\cdotp^{\cdotp^{\cdotp^2}}}}} {\Bigr\}\scriptstyle k},
\ \displaystyle{f(x+\Delta x)-f(x)\over\Delta x}
            \to f'(x){{\rm\ as\ } \Delta x\to0}.
```

Formule 4

```
{{\displaystyle0\thinspace \longrightarrow \thinspace {\cal O}_c
               \thinspace{\scriptstyle\iota\atop \longrightarrow}
 }
 \atop
 { \Big\parallel
   \atop
   {\displaystyle 0 \thinspace\longrightarrow\thinspace{\cal O}_C}
    \thinspace \longrightarrow
 }
}\thinspace

{\rm  ( "gestopt,\ te\ moeilijk" )}
```

6.3 De LaTeX formules

Formule 0

```
${\bf S^{\rm-1}TS=dg}(\omega_1,\ldots,\omega_n)=\bf\Lambda$
```

Formule 1

```
\[ \sum_{i=1}^{\infty}{\frac{1}{2^i}}=1,\ %
   \{\underbrace{\overbrace{\mathstrut a,\ldots,a}^{k\;a'\rm s},
                \overbrace{\mathstrut b,\ldots,b}^{l\;b'\rm s}
               }_{k+l\rm\;elements}
   \},  \ \sqrt[3]{1+\sqrt{1+\sqrt{1+x}}},
\]
```

Formule 2

```
\[ A=\left( \begin{array}{ccc}  a_{11}&\ldots&a_{1n}\\
                                \vdots&\ddots&\vdots\\
                                a_{m1}&\ldots&a_{mn}
            \end{array}
     \right),
   \ \left(\int_{-\infty}^\infty e^{-x^2}\,dx\right)^2=\pi,
\]
```

Formule 3

```
\[ 2\uparr\uparrow k\stackrel{\rm def}{=}
   2^{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}
   \vbox{\hbox{$\Big\}\scriptstyle k$}\kern0pt},
   \ \frac{f(x+\Delta x)-f(x)}{\Delta x}\rightarrow f'(x)\;
                 \mbox{as $\Delta x\rightarrow0.$}.
\]
```

Formule 4

```
\newcommand{\mapdown}[1]{\mbox{$\phantom{\scriptstyle#1}\Big\downarrow$}
                         \mbox{$\scriptstyle#1$}}
\newcommand{\mapright}[1]{\stackrel{#1}{\displaystyle\longrightarrow}}
  \[ \begin{array}{ccccccccc}
              &&&&&&0                                                          \\
              &&&&&&\mapdown{}                                                 \\
             0&\mapright{}       & {\cal O}_c & \mapright\iota & {\cal E} &
               \mapright{\rho}   & {\cal L}   & \mapright{}    & 0           \\
              && \Big\Vert && \mapdown{\phi} && \mapdown{\psi}               \\
             0&\mapright{}       & {\cal O}_C & \mapright{} & \pi_*{\cal O}_D &
               \mapright{\delta} & R^1 f_*{\cal O}_V(-D) & \mapright{} & 0 \\
              &&&&&& \mapdown{\theta_i\otimes\gamma^{-1}}                     \\
              &&&&&& \makebox[0cm]{$R^1f_*\left({\cal O}_V(-iM)\right)
                        \otimes\gamma^{-1}$}\\
              &&&&&& \mapdown{}                                              \\
              &&&&&& 0                                                       \\
     \end{array}
  \]
```

7 Konklusie

- Voor eenvoudige formules en niet al te hoge eisen voor wat betreft de typesetting (b.v. voor diktaten) is WP goed te gebruiken.
- De huidige versie van DECwrite is *niet* geschikt om formules te maken.
- LaTeX geeft de beste resultaten als het gaat om typesetting en mogelijkheden.

Werkgroep 4: Fonts
Hoe maak ik van één font twee fonts?

Erik-Jan Vens

Het is vandaag woensdag 24 juni, de verjaardag van mijn nichtje.

Abstract

Andrea de Leeuw van Weenen wilde een deel van haar zelfgemaakte font aan kunnen spreken met de gewone letters op het toetsenbord, en niet met lastig in te typen en (evt.) lastig te onthouden macros. De aangewezen manier leek het maken van een virtueel font. Dus heb ik eens op een rijtje gezet wat je daarvoor nodig hebt.

Het probleem

Meestal wordt het virtuele font schema gebruikt om van twee fonts één te maken. Maar andersom kan natuurlijk ook. Stel dat we van de ons allen welbekende Computer Modern letter CMR10 twee sets fonts willen maken. Eén font bevat alleen bovenkast, en één font bevat alleen onderkast. En nu zouden we graag zowel hoofdletters als kleine letters willen kunnen typen, maar afhankelijk van het font alleen boven- of onderkast op papier krijgen.

De oplossing

Maak een leesbare beschrijving van het font

De TEX familie bevat een programma `tftopl` waarmee een `.tfm` file (een TEX Font Metric) file omgezet wordt in een 'property list', een `.pl` file. Deze property list file kunnen we met een gewone editor bekijken. We maken dus een file `cmr10.pl` alsvolgt:

```
tftopl /usr/local/lib/tex/fonts/tfm ↩
      ↪ /cmr10.tfm cmr10.pl
```

Het pad naar de file `cmr10.tfm` is natuurlijk operating system en installatie-afhankelijk, maar het doel is in alle gevallen gelijk: een property list file. Deze ziet er ongeveer zo uit:

```
(FAMILY CMR)
(FACE O 352)
(CODINGSCHEME TEX TEXT)
(DESIGNSIZE R 10.0)
(COMMENT DESIGNSIZE IS IN POINTS)
(COMMENT OTHER SIZES ARE MULTIPLES)
(COMMENT             OF DESIGNSIZE)
(CHECKSUM O 11374260171)
(FONTDIMEN
   (SLANT R 0.0)
   (SPACE R 0.333334)
   (STRETCH R 0.166667)
   (SHRINK R 0.111112)
   (XHEIGHT R 0.430555)
   (QUAD R 1.000003)
   (EXTRASPACE R 0.111112)
   )
(LIGTABLE
   (LABEL O 40)
   (KRN C l R -0.277779)
   (KRN C L R -0.319446)
   (STOP)
[...] stuff deleted
   )
(CHARACTER O 0
   (CHARWD R 0.625002)
   (CHARHT R 0.683332)
   )
[...] stuff deleted
(CHARACTER C A
   (CHARWD R 0.750002)
   (CHARHT R 0.683332)
   (COMMENT
      (KRN C t R -0.027779)
      (KRN C C R -0.027779)
      (KRN C O R -0.027779)
      (KRN C G R -0.027779)
      (KRN C U R -0.027779)
      (KRN C Q R -0.027779)
      (KRN C T R -0.083334)
      (KRN C Y R -0.083334)
      (KRN C V R -0.111112)
      (KRN C W R -0.111112)
      )
   )
[...] stuff deleted
(CHARACTER C a
   (CHARWD R 0.500002)
   (CHARHT R 0.430555)
   (COMMENT
      (KRN C v R -0.027779)
      (KRN C j R 0.055555)
      (KRN C y R -0.027779)
```

```
      (KRN C w R -0.027779)
      )
   )
[...] stuff deleted
```

Maar een leesbare beschrijving van je eigen font

We copieëren nu deze beschrijving naar twee files. De één noemen we `cmr10u.vpl`, en de andere `cmr10l.vpl`. Zo'n `.vpl` file heet een 'virtual property list'. Deze file bevat een uitbreiding van de commando's in een property list file. We zullen hier niet alle commando's van de `.pl` file bespreken.

We maken onze twee fonts in een aantal stappen:

1. Maak een titel van je font:
   `(VTITLE CMR10L l.c. characters only.)`
2. Onthoud de checksum en zet 'm op nul.
   ```
   (COMMENT edited checksum:)
   (COMMENT CHECKSUM O 11374260171)
   (CHECKSUM O 0)
   ```
3. Introduceer het *echte* font. Dit is het font dat door de dvi driver gebruikt wordt om te printen. De hier ingevoerde `FONTCHECKSUM` is gelijk aan de zoeven vóór in de file verwijderde checksum.
   ```
   (MAPFONT D 0
      (FONTNAME cmr10)
      (FONTCHECKSUM O 11374260171)
      )
   ```
4. Haal nu de truuks uit die je uithalen wilt. In ons geval willen we dus alleen onderkast op papier krijgen. Dat betekent dat de definitie van (bijv.) de letter 'A' moet zijn: het op papier zetten van de letter 'a'.
   De letter 'A' is alsvolgt in de `.vpl` file beschreven:
   ```
   (CHARACTER C A
      (CHARWD R 0.750002)
      (CHARHT R 0.683332)
      (COMMENT
         (KRN C t R -0.027779)
         (KRN C C R -0.027779)
         (KRN C O R -0.027779)
         (KRN C G R -0.027779)
         (KRN C U R -0.027779)
         (KRN C Q R -0.027779)
         (KRN C T R -0.083334)
         (KRN C Y R -0.083334)
         (KRN C V R -0.111112)
         (KRN C W R -0.111112)
         )
      )
   ```
   We verwijderen de overtollige informatie nu, totdat we alleen dit overhouden:
   ```
   (CHARACTER C A
      )
   ```
   Dan zoeken we de letter 'a' op en pakken daar de breedte, hoogte en diepte van:
   ```
   (CHARACTER C a
      (CHARWD R 0.500002)
      (CHARHT R 0.430555)
      (COMMENT
         (KRN C v R -0.027779)
         (KRN C j R 0.055555)
         (KRN C y R -0.027779)
         (KRN C w R -0.027779)
         )
      )
   ```
   Het metriek gedeelte van deze informatie plakken we bij de beschrijving van de letter 'A'.
   ```
   (CHARACTER C A
      (CHARWD R 0.500002)
      (CHARHT R 0.430555)
      )
   ```
   Nu hebben we een letter 'A' die dezelfde hoogte, breedte en diepte heeft als de letter 'a'. Voor TEX zou het dus klaar kunnen zijn. De driver daarentegen is niet in namen geïnteresseerd, rugnummers moet-ie hebben. Daarom beelden we dit character alsvolgt af op het character 'a'.
   ```
   (CHARACTER C A
      (CHARWD R 0.500002)
      (CHARHT R 0.430555)
      (MAP
         (SETCHAR C a)
         )
      )
   ```

Deze zelfde procedure halen we uit voor `cmr10u.vpl`. Alleen beelden we nu de kleine letters af op het de bovenkast beschrijvingen. Wanneer je ander truuks uit wilt halen, kan dat natuurlijk ook. Je kunt bijv. alle characters vervangen door de 'ff'-ligatuur (dat is octaal character 13).

```
(CHARACTER O 170
   (CHARWD R 0.500002)
   (CHARHT R 0.430555)
   (MAP
      (SETCHAR O 13)
      )
   )
```

Maak een onleesbare beschrijving van je font

Nu hebben we twee files `cmr10l.vpl` en `cmr10u.vpl`. Hiervan maken we `.vf` en `.tfm` files met het programma `vptovf`.

```
vptovf cmr10l.vpl cmr10l.vf cmr10l.tfm
```

Installeer de `.vf` en `.tfm` files in de juiste directory en test of ze werken met `tex testfont`. In het eerste geval moet je een bARBARA bEETON achtige tekst krijgen, in het tweede geval een CDC Cyber achtige tekst.

7^{th} European TEX Conference: EuroTEX '92

September 14–18, Prague

Kees van der Laan

Highlights

- Graphics via TEX and Metafont interaction
- MetaPost: Metafont with Postscript output
- Postscript fonts coupled to TEX
- What every advisory service should know
- AsTEX a model for a scientific workbench
- Tutorials: Advanced TEX, Virtual fonts, XY-PiC, all for free!

1 Introduction

When announced at Paris last year some considered the organization in a Mideuropean country ambitious, especially because rumour has it that companies would not be interested. CSTuG proved us wrong: a very good conference in a nice atmosphere, worth its money and time spent. There was ample time and room for the invited speakers—each morning two of them—and ample time to meet other participants and discuss various issues. Social events were also well looked after: The Black Theatre, Laterna Magica, an organ concert, a visit to the oldest university of Europe, a visit to a monastery, next to the special program for the spouses.

Handy was the public transport passe-partout and the map of the city,[1] especially because the lecture room and the lodging place were some miles apart. The special morning bus assured that the lecture room was well-filled, each morning.

Thanks to grants from DANTE and GUTenberg delegates from several Mid- and Easteuropean countries could participate.

Outside the well-equipped lecture room[2] there were a display with TEX/ Metafont related books, a display with work done via TEX, or Metafont, and a computer room with also e-mail and FTP facilities, so that participants could read their e-mail and exchange files.

Thanks Jiri, Jiri, Karel and colleagues.

Well-done!

In total ≈ 200 people attended the meeting.

The proceedings[3] are (again) incomplete, and not of top quality, alas. Quite a number of typos and other elementary oversights which have nothing to do with TEX, just correct use of the language and a little proofing, that is all.[4] It would have been more pleasant reading if an English speaking TEXie had gone through it. An article in French does not serve a broad audience either, the more so when it is just text, no diagrams nor illustrations to grasp the main points. There is no index. The table of contents contains wrong page numbers, and an important article—A way to ensure the future of TEX: make its use easier on low-cost machines—is not mentioned in the the list. This proceedings would not have passed the barrier of a publishing house, I guess. The contents is good, though, and some papers are really excellent. Form vs. contents, it is just a pity that the form lacks behind the contents. It is a bit peculiar that the two (invited) papers of Frank Mittelbach and Chris Rowley are missing.[5]

From The Netherlands we had the speakers: Harry Gaylord (paper not in the proceedings), Theo Jurriens (only an abstract in the proceedings), Erik-Jan Vens and myself, next to three other Dutch participants. And believe it or not again a Dutchie who had not yet heard of NTG.[6]

Jackowski won the Cathy Booth award for the best paper.[7]

Next year the TUG annual meeting will be in Europe, Aston UK, July 93, with Peter Abbott as head of the organization committee.

There was again no 'Euro-summit.'

In the sequel I will not follow the day-to-day events, but concentrate on main issues.

[1] That the names have been changed of various street in the mean time, did give rise to hilarious stories of TEXies ending up in the middle of nowhere.

[2] With seats for hackers left, users right and those in doubt in the middle, :-)

[3] A copy costs only DM 30,–. See elsewhere in this MAPS (bijlage $\mathcal{G}$). It is worth its money!

[4] I guess this results from us being all volunteers, and doing these things at night, next to the job for living.

[5] Copyright problems I guess.

[6] This proves that our PR activities are not yet good enough, if ever.

[7] He considered it a bit peculiar, because the previous nearly similar paper was awarded at Karlsruhe, three years earlier.

I did travel by car, and visited GUST prior to the meeting.[8]

2 User Group Issues

The traditional European user groups are thriving, although Italic was absent. No participants from Italy, so I wondered whether there is still TEX activity overthere.[9] Yugoslavia was again absent for obvious reasons. CSTuG has a solid base of quite a number of active people and some 200 members. GUST has organized itself with already ≈ 100 members. CyrTuG, and SibTuG, need still some time to get off. HunTuG is modest, and Rumania and Bulgaria just started. It is rumoured that CyrTuG thinks of organizing a meeting for East- and Mideuropean TEXies next year. I think this is a very good idea, because the costs of attending a TUG meeting are prohibitively high, especially for participants from those countries, even if it is in Europe. Furthermore, only delegates might receive a grant, while with a meeting organized in Eastern Europe the threshold for all those interested will be lowered.

There was only time scheduled for the Mid- and Eastеuropean groups to report about their activities and inspire each other with their plans. Good, but not good enough. I would have liked to hear about plans from DANTE and GUTenberg, en public. I dropped recent MAPS-s and David Salomon's courseware for inspection.

3 Presentations

3.1 Graphics

We are aware of the de facto standard way to include graphics via encapsulated Postscript. No papers on that, however.
I found the more or less mature work of Alan Hoenig quite interesting: When TEX and Metafont work together. Very good work and especially worthwhile for handling mathematical graphics via Metafont and incorporating this via a font in TEX. Alan stresses

> 'Metafont is strong in drawing and doing calculations, while TEX is strong in typesetting and file communication.
>
> The idea behind having TEX and Metafont work together is to have each do the things each is good at, and communicate these results to the other.'

His examples are nice and fun, especially the illustrations included in the post-proceedings paper. Captivating is his example of text flowing around irregular shapes. His macros are in the public domain. His paper is bound to be included in some future MAPS.

John Hobby reported about his work with respect to MetaPost, a new language similar to Metafont but with Postscript output, and some features added.[10] His interpreter implements at least the UNIX PIC and GRAP functionalities. His work is available for educational institutions under the non-disclosure agreement. For his paper see elsewhere in this MAPS.

Kristoffer Rose made his work on pretty arrows go public. His approach is classical and similar to LATEX and LaMSTEX. Some discrete line and arrow elements in a font as basics, and macros for the required discrete functionalities. It is available under the GNU general public license. His user manual is also in the proceedings.

Erik-Jan Vens reported about inclusion of Postscript fonts into TEX, giving the TEX community access to the variety of fonts in use by professionals. See for his paper elsewhere in this MAPS.

3.2 Support

A very nice paper was Anita Hoover's 'The key to successful support: knowing your TEX and LATEX users.' An excellent survey of issues people in the support departments should be aware of. Her sloagan is

> 'Never underestimate
> nor overestimate your user.'

Good, very good, excellent! See elsewhere in this MAPS.

Theo Jurriens talked about his experience in using and teaching LATEX in a multi-tool environment. His point is that in such an environment LATEX can be used by everybody. See elsewhere in this MAPS.

3.3 Language aspects

Quite a few presentations dealt with the use of TEX in languages different from English (or better the latin alphabet): Polish, Russian, Arab, and Hebrew. Because I don't speak any of these languages I can't go into detail. Interesting and related is Siebenmann's paper about 'Hyphenation in the presence of accents and diacritics.'

3.4 Future of TEX

Phil Taylor presented an excellent survey of the New Typesetting System activities. This paper or its successor is bound to be redistributed within MAPS in the near future.

Frank Mittelbach and Chris Rowley's 'In pursuit of Quality,' had a weak basis: narrow columns of text

[8] I was happy to see the Poles rebuild their society, many people building new houses, and I became a little sad about the situation in the former DDR, although the reconstruction of roads has started.

[9] Consulting the TUG resource directory soothed me. It lists some 17 cities with TUG members in Italy.

[10] He reported already about it at Stanford, 1989, TUGboat 10, 4, 505–512. The user manual is in the AT&T Bell Laboratories report series: CSTR 162, April 1992.

gives rise to difficulties. A revisiting of Richard Southall's 'Buses and weirdness,' Groningen 1990. This is precisely the reason why professional journals don't use 2-column format. AMS (and also the British Math society, I was told) gain awards for outstanding typography attained at via TEX. So, Frank's talk was too negative, and one-sided, only looking at the bad practice. He should have started with the good discipline, and comment on that. The message is true however: don't use multi-column output for automatic high-quality work, unless you are willing to do too much post-proof adaptation.

Very intriguing is Michel Lavaud's mature paper about again As(sisted)TEX, his Scientific Document Processor environment. I completely agree with his message

> 'Make the use of TEX easier on low-cost machines.'

But, that is only part of the story: education is also very important, next to good PD versions.
Basically, he envisions that the near-future researcher will start with his document, and sidestep for doing experiments or simulations. I day-dreamed about this some years ago. It is opposed to the current practice of doing first the (thesis) research and then lock yourself up for the writing. With this in mind he has built his AsTEX. With respect to the complexity of AsTEX, see my discussion of it in the EuroTEX '91 report.
This paper taught me a sensible nowadays and near-future way of TEX-ing on low cost PC-s. The ingredients to make that possible are

> front-end (framework and desqview)
> TEX, and
> backend (a previewer with zoom and scroll functionality).

This all supported by multitasking and interprocess communication.

> 'Using TEX on a 80386 notebook with OS/2 and emTEX is certainly a better choice than using it on a workstation, because this cuts price and not performance, and the machine can be carried anywhere.'

Hummmm, . . . sounds good, a demonstration would be nice. Next to Blue Sky's Lightening TEX on an 'Apple.' At an NTG meeting? Next June?

Laurent Siebenmann in his 'The lion and the mouse,' argued in the same direction. Macintosh-like user interface (the 'mouse') should be common practice, also in relation with TEX (the 'lion'). His product is called LinoTEX. He tendered some challenges with respect to the 'mouse ⇒ lion' interaction (in that direction). Also Lavaud pointed out that this backward interaction—from the preview window smoothly and fast into the edit window at the right spot—would speed up TEXing. According to Laurent the milestones for interactive TEXing are

1986 David Fuchs' page-by-page preview

? Eberhard Mattes' word search in the preview (rather than source)

1991 Blue Sky's continuous output via Lightening TEXtures.

3.5 Utilities.

On this front there was: Taupin with his MusicTEX (see MAPS92.1), Rose with his XY-PiC (GNU PD), for typesetting pretty (commutative) diagrams, and my bordered table macro (see elsewhere in this MAPS) as well as my crossword macro (see elsewhere in this MAPS). My separation of the data specification from the typesetting earned some appraisal: '. . . it is so simple and natural.'

3.6 Publishing houses

Interesting was the presentation of MIR. They translated and typeset Concrete Mathematics, and the Joy of TEX into Russian.

Vacha reported about the experiences with TEX in publishing houses in Czecho-Slowakia.

At nearly every TEX-conference I attended, there was a report about TEX as formatter coupled to a database. Petr Sojka, Rudolf Cervenka, and Martin Svoboda (excuse me, neglecting the accents) reported about their experiences with the formatting by TEX of Terminological Data Bank output, created by the CDS/Isis public domain database tool.

3.7 Education

Not much about education, really. Too bad.

3.8 LATEX 3

Again no dates of yet. The big news for me was the appraisal, en public, for TEX's kernel being frozen!

4 Tutorials

There was not a Q-&-A-session as such. However, Phil Taylor started his tutorial by composing a list of items to be discussed by consulting the audience about what they would like to be treated. Quite impressive, and courageous, although one can predict with near-certainty the main issues. Even more impressive was that he taught all by head! One can always learn something when attending TEX classes: Do you know for example the difference between `\aftergroup` and `\afterassignment`? Perhaps you don't have to, because in applications it comes out quite natural.[11]

[11] The difference is that `\aftergroup` can store more than one token and `\afterassignment` only one, the most recent one.

Another eye-opener was the reason why sometimes an explicit opening brace is needed as opposed to `\bgroup`.[12] Phil Taylor dropped during his tutorial that he would be happy to receive challenging typesetting problems: P.Taylor@vax.rhbnc.ac.uk.

Other tutorials were by Yannis Haralambous about virtual fonts, and by Kristoffer Rose about his XY-PiC. All collateral.

5 Parting gift

At the end of the conference Karel Horak and Petr Olsak, forgive me the lack of diacritical marks, surprised everybody with their nice handout: 'The pamphlet on TEX fonts,' sized a quarter A4. And of course there was the proceedings on Friday.

Once again, thanks to you all!

NTG's copy of the proceedings will be available for inspection at the readers' table at NTG's fall meeting. If the authors will grant permission, I will bring for inspection a proof of 'The LATEX Companion'-book, by Michiel Goossens, Frank Mittlebach, and Alexander Samarin, to be published by Addison-Wesley.

[12] When the `\bgroup` is ambiguous, and can be seen as a parameter separator.

Verslag van de TUG conferentie in Portland, Oregon

Johannes Braams

september 1992

1 Inleiding

Dit jaar had ik het genoegen de TUG conferentie in Portland, Oregon bij te kunnen wonen. Samen met een collega had ik een 'abstract' voor een paper ingestuurd en die was geaccepteerd door het programma commitee. Omdat ik pas relatief laat de reis boekte, was ik 'gedwongen' twee weken vakantie vooraan de conferentie te koppelen. In die twee weken ben ik, samen met mijn vrouw, van Los Angeles naar Portland gereden met de auto, een mooie tocht.

De conferentie begon op 27 juli met een welkomsreceptie. Dergelijke recepties zijn altijd goede gelegenheiden om kennis te maken met mensen waarvan je wel het e-mail adres, maar niet het gezicht kent. Ook worden er afspraken gemaakt voor gesprekken later in de week.

2 Maandag 28 juli

De conferentie begint voor ons, sprekers al vroeg. Om 7:00 uur is een sprekers-ontbijt georganiseerd, waar de sprekers instructie zullen krijgen aan welke regels ze zich moeten houden als ze 'op het podium' staan. Het blijkt meer ontbijt dan instructie, maar dat is ook gezellig.

Na enkele huishoudelijke mededelingen wordt het spits afgebeten door Malcolm Clark, de voorzitter van TUG. Malcolm weet op zijn bekende wijze het publiek bezig te houden. De boodschap die hij brengt is dat wij TEXies te vaak weten te vertellen wat er allemaal niet deugt aan TEX, terwijl er te weinig verteld wordt wat TEX allemaal *wel* kan. Een voorbeeld hiervan is één van Malcolms stokpaardjes, de kombinatie van TEX en 'graphics'. In plaats van op te merken dat TEX er niet goed in is zou dit ook anders benaderd kunnen worden, Malcolm is „stunned how well TEX does do graphics". Er wordt volgens Malcolm te veel aandacht besteed aan de 1% van de problemen die TEX niet aan kan. De nadruk zou meer moeten liggen op de „portability" en „archivability" van de documenten die met TEX gemaakt worden.

In de loop van zijn betoog wordt zijn verhaal steeds meer een persoonlijke mening; speciaal wanneer hij het heeft over de vraag of er een opvolger van TEX moet komen. Zijn conclusie: laat TEX zoals het is en besteed aandacht aan de integratie van TEX's omgeving, de cyclus edit – TEX – preview – print.

De verdere ochtend is gewijd aan 'graphics and TEX'. De eerste spreker is David Salomon die rechthoeken van een schaduw voorziet door een TEX macro met argumenten te combineren met een stukje PostScript code. „It's very simple, easy to use, easy to modify" aldus David.

Robert Harris vertelt ons vervolgens over zijn ervaringen met het opnemen van foto's in TEX documenten. Hij maakt jaarboeken voor een hondenvereniging en moet daarin foto's van honden opnemen. Zijn resultaten met de huidige mogelijkheden, scanners en software, zijn niet slecht.

De ochtend wordt besloten door Jackie Damrau die het heeft over „Discovering Graphics in LATEX Documents". In haar organisatie worden veel handboeken geschreven, waarin regelmatig tekeningen van verschillende aard moeten worden opgenomen. De meeste zijn gemaakt op een MacIntosh en worden in PostScript formaat met behulp van een `\special` in een document opgenomen. Dit alles op verschillende platforms (Unix, VMS, Mac) en met verschillende PostScript vertalers (DVIlaser/PS, T2/Script, *Textures*).

Na de zogenaamde 'networking lunch', waarin men geacht wordt te discussiëren over een onderwerp dat op een briefje op de tafel ligt, gaat de dag verder met een verhaal van Neil Weiss getiteld „Incorporation of PostScript Figures using MG". Of dat iets met een befaamd automerk te maken heeft kan ik niet zeggen, omdat ik die tijd heb gebruikt voor een overleg met Ron Whitney over gecombineerde lidmaatschappen NTG/TUG.

De tweede voordracht wordt verzorgd door Victor Eijkhout. Hij licht een aantal tipjes op van de lollipopsluier. Met lollipop heeft Victor een poging gedaan het interface naar TEX in drie niveau's te splitsen, elk voor een andere doelgroep. De onderste laag is TEX zelf, voor de TEX-programmeur. Daarin heeft hij een taal geïmplementeerd die bedoeld is voor de document ontwerper. Die definiëert vervolgens het interface voor de gebruiker, het derde niveau. Aan de hand van een aantal voorbeelden laat hij zien hoe dit in elkaar steekt. Het lollipop formaat is niet af; zo is er geen ondersteuning voor tabellen en tekeningen, de ondersteuning voor wiskunde is beperkt. Victor vraagt om vrijwilligers die hem moeten helpen lollipop af te maken. Wat

hij heeft zal via ftp beschikbaar gesteld worden[1], het adres zal in *TUGboat* gepubliceerd worden.

Voor de theepauze krijgen een aantal leveranciers de gelegenheid hun kunnen te tonen. De eerste is Barry Smith van Blue Sky Research. Hij geeft een zeer indrukwekkende demonstratie van hun nieuwste produkt, 'Lightning *Textures*'. In één window wordt de invoer ingetypt, terwijl in een ander window het resultaat direkt te zien is. Bij iedere toetsaanslag wordt het hele document opnieuw door TEX bewerkt. Als indicatie van de snelheid van deze versie van TEX vertelt Barry dat *The TEXbook* in 2 minuten verwerkt wordt.

Hierna neem Betsy Dale van Arbortext het woord om iets te vertellen over hun SGML implementatie op basis van TEX. Robert Harris van Micro Programs, een Arbortext distributeur, krijgt als laatste de kans iets over zijn bedrijf te vertellen.

Na de thee wordt een panelsessie gewijd aan graphics in TEX. Er wordt daar door een aantal mensen vloeiend PostScript gesproken. Eén van de discussie onderwerpen was het opnemen van kleur in TEX-documenten.

De TEXnisch inhoudelijke kant van de dag wordt afgesloten met een aantal workshops. Door mij werd de „Archive and File Management" workshop, geleid door Peter Abbot bijgewoond. Peter beschrijft de situatie in Aston met betrekking tot het archief en geeft enig inzicht in de plannen die leven. Zo komt er in Aston en public access 'gopher'. Een belangrijk resultaat van de bijeenkomst is dat wordt afgesproken ernaar te streven dat *alle* ASCII files in de archiven voorzien worden van de standaard headers, zoals voorgesteld door Nelson Beebe. Hiertoe zal Barbara Beeton kontakt opnemen met Don Knuth om ook de `plain`TEX en computer modern sources hiervan te voorzien.

De dag wordt tenslotte besloten met een wijn en kaas receptie die door Personal TEX, Inc wordt gesponsord.

3 Dinsdag 29 juli

De dinsdag start gelukkig niet zo vroeg als de maandag. Volgens het programma komen eerst bedrijven aan bod die in de tentoonstellingsruimte hun kunnen laten zien. Aan het woord komen achtereenvolgens:

- Mimi Jett van ETP, een bedrijf dat zich bezig houd met het produceren van boeken en tijdschriften en daarbij vooral met TEX en LATEX werken;
- Ami Hendrickson van TEXnology Inc. Zij laat wat voorbeelden zien uit haar eigen macropakket, macro TEX. Het pakket is nog steeds in ontwikkeling en zij wil te veel in te weinig tijd laten zien.
- Berthold Horn van Y & Y. Hij heeft samen met Bigelow and Holmes gewerkt aan de ontwikkeling van het lettertype Lucida Bright. Zijn missie is „TEX without bitmaps". Hij noemt nog de programma's van DVIPSONE dat een deel van TYPE 1 outline fonts kan 'downloaden', en DVIWINDO, een previewer voor MS-Windows die zowel met TYPE 1 als met TRUETYPE fonts overweg kan.

De eerste echte lezing van de dag wordt verzorgd door Bart Childs. De titel van zijn voordracht is „Literate Programming: A Practitioners (biased) View". Volgens hem heeft Knuth met het WEB-systeem wellicht een groter bijdrage aan de wereld geleverd dan met TEX. Hij gaat in op de vraag wat een goed WEB-programma is aan de hand van grafieken waarin de verhouding tussen het aantal regels code en het aantal regels commentaar wordt gegeven voor een aantal programma's.

De tweede lezing van de ochtend is gewijd aan het converteren van documenten. Dennis Arnon stelt een aanpak voor via een tussenstap. Hij gebruikt hiervoor een 'genre model', gebaseerd op de stelling:

> Families of attributed trees are good datastructures for modelling document genres.

Na een theoretisch betoog over contextvrije en en phylum/operator grammatica's laat hij aan de hand van een uitgereikt voorbeeld het resultaat zien van een programma dat op basis van zijn theorie werkt. Het is duidelijk dat hij nog wat werk te doen heeft.

Na de koffie mag Barry Smith meer vertellen over „A High-Performance TEX for the Motorola 68000 Processor Family". Voor het produkt 'Lightning *Textures*' had Blue Sky Research een versie van TEX nodig met een zo groot mogelijke snelheid. Om dat te bereiken is men ertoe over gegaan TEX in 68000 assembler te implementeren. De assembly-code uit de PASCAL compiler is aan de hand van een aantal profiling hulpmiddelen geoptimaliseerd. Het resultaat is dat TEX drie maal zo snel is geworden en $1/3$ van het aantal instructies nodig heeft om het *The TEXbook* te bewerken.

Het meest verrassende verhaal van de conferentie was dat van T.V. Raman, getiteld „An Audio View of TEX documents". Zijn verhaal is een gevolg van zijn PhD opdracht, waarin onderzoek doet naar methoden om LATEX documenten in gesproken tekst om te zetten. Zijn interesse in dit onderwerp is waarschijnlijk een gevolg van het feit dat Raman blind is. Hij werkt momenteel aan een twee-staps methode met een model op hoog abstractie niveau van wat een document is. Hij legt uit, nog problemen te hebben met een mengsel van structurele en vormgevingscommando's. Het basis probleem volgens hem is dat een gedrukt document twee-dimensionaal is terwijl audio maar een dimensie kent. Vooral bij wiskundige formules komt dit probleem om de hoek kijken. Uit zijn demonstratie blijkt dat hij het gebruik van verschillende fonts vertaalt door het gebruik van verschillende stemmen. Verschillende elementen van een document worden van elkaar onderscheiden door verschillende toontjes.

[1] Dat is inmiddels gebeurd, lollipop is beschikbaar op cs.utk.edu maar ook op LISTSERV@HEARN.BITNET

De laatste lezing voor de lunch is van Peter Abbott, die vertelt hoe hij het verenigingsblad van een locale historische spoorweg 'The Cornishman' maakt. Voorheen kwam het blad nooit op tijd uit; sinds Peter het met LaTeX maakt is de kwaliteit verbeterd en komt het blad op de geplande datum beschikbaar. Volgens Peter geldt: „TeX is a system for setting beautiful text."

De middag wordt gevuld met een panelsessie over TeX archieven en een TUG business meeting. De TUG conferentie zal in 1993 in Aston (Groot Britannië) worden gehouden. De datum ligt nog niet vast, maar eind juli is het meest waarschijnlijk. Het thema van de conferentie zal zijn „World Wide TeX".
Van het financiële front wordt gemeld dat TUG in 1991 een verlies heeft geleden van $ 6.000,– (dat is kleiner dan men gevreesd heeft). Echter door het herwaarderen van voorraden is het negatieve eindresultaat $ 25.000,–. Zoals het er op dat moment uit ziet zal de begroting voor 1992 worden gehaald. De contributie zal daardoor niet nog een keer verhoogd hoeven te worden. Om het TUG lidmaatschap voor studenten aantrekkelijker te maken is besloten het studentlidmaatschap van TUG voor 50% van de prijs van een 'normaal' lidmaatschap aan te bieden. Door verschillende oorzaken is het ledental dit jaar gedaald tot 3200.
TUG is op zoek naar zowel een nieuwe 'executive director' als naar een nieuwe voorzitter; Malcolm Clark zal per 31 december af treden.
Het bestuur meldt ook dat op dit moment hard gewerkt wordt aan een nieuwe versie van de 'bylaws' (statuten). Daarin wordt ook een nieuwe verkiezingsprocedure voorgesteld: de verkiezingen vinden plaats tijdens een TUG bijeenkomst en ieder jaar wordt de *helft* van het bestuur opnieuw verkozen. Om dit te kunnen bereiken wordt voorgesteld de zittingstermijn van de nieuwe voorzitter $2\frac{1}{2}$ jaar te laten zijn, terwijl de helft van de 'board' die volgend jaar verkozen wordt vanaf 1 januari 1994 een zittingstermijn heeft van $1\frac{1}{2}$ jaar, terwijl de anderen $2\frac{1}{2}$ jaar zitting dienen te nemen.

Na afloop van de vergadering hield Blue Sky Research open huis, alle conferentiedeelnemers waren welkom. Ze blijken een mooie verzameling boeken over typografie, boekdrukkunst en lettertypes te bezitten. Na nogmaals een demonstratie van lightning *Textures*, is iedereen het er over eens dat het zonde is dat dit alleen op de MacIntosh beschikbaar is.

De dag wordt besloten met een 'social event', een diner op de hoogste verdieping van een flinke flat. Er is een fraai uitzicht op Portland en de vulkanen (Mount Hood en Mount St. Helens) in de direkte omgeving.

4 Woensdag 30 juli

Net als dinsdag begint de dag met presentaties van 'exhibitors'.

- Als eerste mag de AMS vertellen welke diensten zij de TeX wereld aanbiedt. Voor de PostScript gebruikers wordt gemeld dat gewerkt wordt aan een versie van de AMS-fonts in Adobe type 1 formaat.
- Don Hosek van „Quixote digital typography" heeft een TeX consultancy bedrijfje. Hij heeft twee 'abonnementen' voor mensen die ondersteuning wensen. Je kunt bij hem voor $2.000,– een kwartaal lang net zo vaak bellen als je wilt of voor $ 400,– twintig telefoontjes plegen.
- Larry Bennet komt vervolgens reclame maken voor T-Edit, een pakket om het gebruik van TeX in een Dos omgeving te vereenvoudigen. Om het te kunnen gebruiken moet je echter wél in het bezit zijn van de KEDIT editor, REXX voor Dos en een kleuren monitor.

De eerste echte lezing wordt verzorgd door Kresten Krab Thorup[2]. Hij presenteert AUCTeX, een LaTeX edit mode voor emacs. Bij de nieuwste release (versie 6) is nu ook documentatie beschikbaar. De omgeving biedt mogelijkheden om met zeer weinig toetsaanslagen een environment toe te voegen, een deel van te tekst weg te commentariëren (of juist het %-teken weg te halen), zorgt voor 'pretty printing' van de ASCII bron van het LaTeX document en nog veel meer. Onderdeel van het pakket is ook het programma `lacheck` dat een 'consistency check' uitvoert op een LaTeX document. Het hele pakket is beschikbaar (via ftp) op iesd.auc.dk (130.225.48.4).

Na de koffiepauze komt Chris Rowley ons onderhouden over het LaTeX3 project. Hij vindt dat het doel waarvoor LaTeX ooit is ontworpen en gebouwd niet helemaal is bereikt. Het LaTeX3 project probeert nu dat doel beter te benaderen. In ieder geval zal er over het resultaat van het project uitgebreide documentatie beschikbaar komen[3]. Chris verklaart het feit dat men tot nu toe heel weinig resultaten uit het project heeft kunnen zien komen. LaTeX3 wordt geheel van de grond af aan opnieuw opgebouwd.

Na Chris komt Yannis Haralambous aan het woord. Zijn onderwerp is „Arab and Hebrew TeX revisited". Zijn stelling is: „TeX's strength is good Arabic typesetting.". Yannis heeft op basis van een boek over arabische boekdrukkunst en TeX–XeT een systeem gebouwd om klinkers *en* accenten correct te plaatsen. Eén van de moeilijke aspecten van arabische typografie is dat er zeer veel ligaturen voorkomen, ook van drie en vier tekens. Dat levert zo zijn eigen problemen op. De resultaten die Yannis laat zien, zijn (zoals altijd bij hem) indrukwekkend.

De ochtend wordt besloten met een lezing van James Hafner over „FoilTeX, a LaTeX-like system For Typesetting Foils". James werkt bij IBM en een 'foil' is IBM-taal voor een 'overhead sheet'. Hij claimt dat zijn

[2] die dinsdag al een optreden had als assistent van T.V. Raman.
[3] In Praag heeft Frank Mittelbach daartoe een overeenkomst met Addison & Wesley gesloten.

oplossing, die gebaseerd is op LaTeX, flexibeler is dan SliTeX. Voor degene die de beschikking heeft over een kleuren PostScript printer bestaat de mogelijkheid kleurige 'foils' te maken.

De middagbijeenkomst begint met een verhaal van Anthony Starks over Dotex, een geïntegreerde omgeving voor TeX in een X-window omgeving. De tweede spreker, Dave Marks, heeft een macro pakket bovenop PostScript ontwikkeld om tekeningen te kunnen maken. Hij voegt daarmee een aantal commando's aan de PostScript taal toe. De voorbeelden die hij laat zien zijn fraai, maar de moeite die het gekost heeft om ze te maken is misschien wel erg groot.

Na de thee wordt de bijeenkomst voortgezet met een panelsessie, getiteld „Font Forum". De zitting begint met een serie inleidingen van de panelleden.

- Berthold Horn vertelt dat 'TrueType' fonts schaalbare, resolutie onafhankelijke 'outline' fonts zijn.
- Arthur Ogawa legt uit dat 'Multiple Master Fonts' langs 4 dimensies interpoleerbaar zijn.
- Yannis Haralambous geeft een overzicht van de beschikbare afbreekpatronen voor de verschillende talen die geschikt zijn voor de DC-fonts en de Cork font-encoding standaard.
- Chris Rowley benadrukt het belang van NFSS[4]. Hij legt uit wat het is, wat er mee kan en dat het echt niet alleen voor LaTeX is.

Tijdens de discussie wordt gemeld dat in 1993 het door Didot-project een METAFONT-'school' wordt georganiseerd. Het doel is het maken van een didot lettertype met METAFONT.

Na afloop van de paneldiscussie zijn een aantal 'Birds of a Feather' bijeenkomsten gepland. Daarvan woon ik de LaTeX3 BOF bij, waarin een lijst met taken voor vrijwilligers wordt uitgedeeld. Het is de bedoeling dat een aantal taken door mensen worden uitgevoerd die niet direkt bij de kern van het project betrokken zijn. De lijst zal ook in *TUGboat* worden gepubliceerd en op de bekende TeX archieven beschikbaar worden gemaakt. Voor elektronische discussie wordt verwezen naar LATEX-L@DHDURZ1.BITNET.

De dag wordt besloten met een 'LaTeX3 fundraiser'. Hiertoe is door Malcolm Clark en Doug Henderson een Bowling toernooi georganiseerd. Per gespeeld spel moet $ 5,– worden betaald, waarvan een groot deel voor het project bestemd is. Zo'n 32 mensen doen mee en hebben een gezellige avond.

5 Donderdag 31 juli

Alweer de laatste dag van de conferentie. De dag begint, net zoals de vorige, met presentaties van 'exhibitors'. Vandaag komen aan het woord:

- Fred Osborne van TCI, het bedrijf dat 'Scientific Word' op de markt brengt. Scientific Word is een soort WYSINQWIG[5] editor voor TeX documenten.
- Ron Whitney en Cliff Alper van het TUG bureau. Ron vertelt dat op 30 september een nieuwe cursus, 'TeX for Publishers' voor het eerst gegeven wordt bij John Wiley & Sons in New York. TUG heeft vrijwilligers nodig en roept die nu op zich te melden. Er is onder meer behoefte aan mensen die TeXnische vragen kunnen beantwoorden, hulp bij de produktie van *TUGboat* en mensen die een TUG bijeenkomst willen organiseren.

De eerste lezing wordt verzorgd door Walter van de Laan en mijzelf. Onze presentatie gaat over de manier waarop bij PTT Research de kwartaal rapporten, die door veel verschillende mensen geschreven worden, samengesteld worden. De oplossing is gebaseerd op een mail-server die ervoor zorgt dat een ASCII text door LaTeX bewerkt kan worden. Het eigenlijke rapport wordt in de batch met behulp van LaTeX gemaakt.

Na de koffiepauze komt Robert McGaffy aan het woord over „SGML, C, and TeX for automatic tables". Hij gaat uit van een SGML beschrijving van tabellen en gebruikt TeX om de inhoud van een tabel te meten. Gegevens over hoogte, breedte en diepte van de cellen en de lengte van de tabel worden in een file opgeslagen. Die informatie wordt vervolgens gebruikt om de tabel zonder `\halign` constructie op papier te krijgen. Wat hij doet ziet er veelbelovend en ingewikkeld uit, maar het is nog niet klaar.

Mimi Burbank vertelt ons vervolgens hoe zij in haar instituut gebruik maakt van TeX voor een databank van publicaties. Uit de databank moeten op een aantal verschillende manieren overzichten gegenereerd worden 'om geldgevers tevreden te houden'.

De laatste spreker voor de lunch, Timo Knuutila heeft als onderwerp „How to Handle Multiple languages and Character Sets with TeX". Hij claimt dat zijn oplossing compatibel is met mijn babel systeem. Hij gaat op één punt een stapje verder dan babel; zijn systeem biedt ook de mogelijkheid voor een andere taal ook een andere fontfamilie te gebruiken. Hij introduceert vervolgens weer eens een uitbreiding op de cm-fonts, door hem xcm gedoopt. Zijn codevector is op ISO Latin-1 encoding gebaseerd.

Na de lunch wordt door de leiders van de BOFs verteld wat zij in hun sessies gedaan hebben. Dan volgt de laatste lezing van de conferentie. Die eer valt te beurt aan Harry Baldwin. Zijn voordracht heeft als titel „Using True Basic as an aid to writing TeX Documents". Hij heeft een uitgebreide databank met wiskundige vraagstukken opgebouwd, verdeeld in tien categoriën van elk 200 gelijkwaardige vragen. Uit die databank stelt

[4] NFSS staat voor 'New Font Selection Scheme' en is door Frank Mittelbach en Rainer Schöpf ontwikkeld en in *TUGboat* gepubliceerd.

[5] What You See Is Not Quite What You Get

hij proefwerken samen met behulp van een True Basic programma dat de vraagstukken voor hem selecteert.

De conferentie wordt besloten met een 'board roundtable'. Vrijwel het voltallige bestuur – niet iedereen is op de conferentie aanwezig – neemt plaats op een podium om vragen van de conferentie deelnemers te beantwoorden. Tijdens deze bijeenkomst wordt bekendgemaakt dat de netto opbrengst van de bowling avond ongeveer $ 750,– bedraagt. Een symbolische cheque wordt namens Frank Mittelbach door Chris Rowley zeer dankbaar in ontvangst genomen.

Formeel is de conferentie nu afgesloten, maar informeel nog niet, want 's avonds is een bijeenkomst in een proeflokaal van een lokale bierbrouwerij georganiseerd. Onder het genot van gratis bier (of iets anders) en een rondleiding door de brouwerij kan uitgebreid nagekaart worden over een succesvolle conferentie.

Nadat we van een aantal mensen afscheid hadden genomen met „Tot Praag!", vlogen we (na nog een dag 'sight seeing') via Canada terug naar Nederland.

The Key to Successful Support: Knowing Your TEX and LATEX Users*

Anita Z. Hoover

University of Delaware
002A Smith Hall
Newark, DE 19716
anita@brahms.udel.edu

Abstract

The primary emphasis of this paper is to address the issues related to supporting TEX and LATEX. One essential ingredient to successfully supporting any package is that you must know your users. In the case of TEX and LATEX, this is especially true, because the user base can be so diverse. This paper will focus on support strategies that address different types of users and what you can do as a TEX and LATEX support person to adopt these strategies in your organization.

Keywords: user support

1 Introduction

When I started working at the University of Delaware, one of my primary responsibilities was to build a sound structure to support TEX and LATEX. Well, five years have passed, and I can say that a good foundation for support is in place but not without a lot of learning and understanding. I started this job never having used TEX or LATEX. Shortly after getting involved, I realized that this was not going to be a short-term project, but one that would last forever. What I mean by this is that from week to week, I would learn of a new macro or style file, new previewer, new printer driver, new utility, new user and kept wondering how I was possibly going to stay above water and offer good support in such a changing environment.

In the preface of the *TEXbook*, I think that Knuth [6] summed up the transition one goes through when learning a powerful tool like TEX. I think his description can be taken a step further in defining the role of a support person—how he or she needs to grow and change in order to provide good support.

> . . . When you first try to use TEX, you'll find that some parts of it are very easy, while other things will take some getting used to. A day or so later, after you have successfully typeset a few pages, you'll be a different person; the concepts that used to bother you will now seem natural, and you'll be able to picture the final results in your mind before it comes out of the machine. But you'll probably run into challenges of a different kind. After another week your perspective will change again, and you'll grow in yet another way; and so on. As years go by, you might become involved with many different kinds of typesetting; and you'll find that your usage of TEX will keep changing as your experience builds. That's the way it is with any powerful tool: There's always more to learn, and there are always better ways to do what you've done before. At every stage in the development you'll want a slightly different sort of manual. You may even want to write one yourself.
> . . .

I think it is important to remember that you will always have users at these different stages of learning and, as such, the type of support you provide will also vary depending on the level of the user. It is also important to keep in mind that you will always have new users that will invariably require special attention when beginning to use TEX. Remember that you were once there yourself, and it is sometimes easy to forget and overlook the level of explanation required for new users. You will find that the time spent with these beginners will pay off in the long term because these users will become the support network in your organization, solving the basic questions within their own department.

With this in mind, I feel there are three areas of support that need to be addressed

*Presented at EuroTEX '92, September 14–18, Prague, Czechoslovakia.

1. Users
   What kind of users are using TEX and LATEX?
2. Style files and macros
   What particular style files and/or macros are necessary to fulfill the user's needs?
3. Tools
   What tools are necessary to provide a good working environment for using TEX and LATEX.

2 Users

Through my experiences, I can break down the different types of users into three groups:

1. Graduate students
2. Professors or professional staff who are writing books and/or journal articles
3. Technical secretaries

Our largest audience for all three groups listed above is in the "sciences." As a result, most of the users had already discovered the beauty of TEX and LATEX. They were convinced of its ability to offer high quality mathematics and overall typographic quality required for their documents.

However, the level and type of support required is quite different for each of these groups.

2.1 Graduate Students

Most graduate students familiarize themselves with the basics of TEX or LATEX, depending on which package what they want to use to write their thesis or dissertation. They usually purchase or have available to them a copy of the appropriate manual, *The TEXbook* [6] or *LATEX: Users Guide and Reference Manual* [8]. They also purchase a *Thesis Manual* [9] from the Office of Graduate Studies explaining the required specifications.

Based on the above analysis, I found that the best support for graduate students was to

1. develop a style file for LATEX and a set of macros for TEX that would meet the requirements of the Office of Graduate Studies at the University of Delaware. The style file and macros are called UDThesis.
2. write a document with examples that explain how to use the UDThesis style file for LATEX [1] and macros for TEX [2].
3. develop a short course (two hours) based on the documentation that would primarily be hands on. The course would focus on the creation of a basic front and body of a thesis or dissertation.

This method has proven to be very effective [5]. Even though graduate students are usually only around for several years, their experiences help considerably with support because they help new graduate students and encourage them to read the documentation and attend the short course. We also work very closely with the Office of Graduate Studies to ensure that our macros meet their specifications, and they too inform us of any problems that arise to determine if the macros are incorrect or the student made a mistake. This is a tremendous help believe it or not, as we have found problems with the macros that must have been there for months, but students just would get around them by using correction fluid or cutting and pasting in changes.

In general, you will begin to see that good support just doesn't come from one person (even if technically you are the only designated support person). It is crucial to be able to communicate and work with others in your organization to provide a successful support network.

2.2 Professors and Professional Staff

Again, this particular group of users is usually familiar with the basics of TEX and LATEX. They also purchase or have available to them a copy of the appropriate manual. However, I have found that two levels of support are necessary for this group based on whether or not they were provided with style files and/or macros to meet the requirements of the publisher [5]. I classify them into two groups:

1. User-defined macros
   The user must define macros to design the document to meet a publisher's specifications and then submit a final printed copy.
2. Publisher-defined macros
   The publisher supplies macros that meet the publication specifications. The user uses these macros to design the document and then either submits a final printed copy or sends in the TEX/LATEX file.

In terms of support, group two is relatively self-sufficient provided the documentation is written well and examples are included. In most cases, the only support required is interpretation of the publisher's terminology and documentation.

Group one, on the other hand, can sometimes be a real challenge to support. Although most of these users are familiar with TEX/LATEX, they are not familiar enough with modifying or creating style files or macros to follow the specifications of a publisher. Currently, I need to work with each user and make the necessary changes. This is the most time-consuming activity as a support person. However, if the information is shared with the user, then more than likely he or she can use this information for future documents. I feel that two manuals are desperately needed: (1) "How to modify LATEX style files" and (2) "How to set up a book/journal specification with TEX." I understand that the first of these two manuals is soon to arrive on the scene.

In both groups, you will find that the key to minimizing support lies in the ability of the author and publisher to communicate. Much time is wasted on issues concerning design and layout. The author says they are unhappy with a particular format, the author asks you to make the changes, the publisher sends back edited copies that change it back to the original specifications, and on and on. This process can get quite frustrat-

ing for the author, publisher and support person. Try to encourage the user to make these issues as clear as possible up front. This kind of support will be more valuable than getting into a design war.

2.3 Technical Secretaries

This particular group of users is not usually familiar with TEX and LATEX. More importantly, they are not accustomed to teaching themselves by using a manual. I have found that many times the problems are not even specific to TEX/LATEX, but more the basic unfamiliarity with the background in which the paper was written, Greek symbols, mathematical expressions, etc. These problems introduce an entirely different level of complexity based on support. Many of these types of problems were stated clearly by Kubek [7]:

> . . . TEX use seems to be largely confined to the technical professionals with degrees in the fields listed above, who have taught themselves with only the *TEXbook* to guide them (and wouldn't have a clue as to how to teach anyone the program who doesn't speak *at least* three programming languages). For those of us who had to look up the word "algorithm" several times in the dictionary and still wouldn't recognize one even if it asked us to dance, a major dilemma arose when faced with the prospect of learning TEX "by the book." Not only is the *TEXbook* oriented toward programming, but a lot of the commands don't even look like they're in English!

Based on similar experiences, I have found the following to be most successful in supporting this group of users:

1. Provide two short courses (three hours each) that teach the basics of TEX/LATEX. This course is mostly hands-on, where a particular formatting technique is discussed, and the participants in the class type in an example using the information they just learned. They process the file using TEX/LATEX and preview it to see their results.
2. Provide exact examples (as many as possible) that cover commonly asked questions like, "I need to change my margins," "I need to change my equation numbering," "I want a full page figure or table," etc.
3. Time permitting, take advantage of some of the common word processor features available to interface with TEX and LATEX and provide some short cuts for typing and error detection. For example, designing keyboard layouts and macros for WordPerfect [4].
4. Encourage users to share their experiences and examples with each other to solve their problems.

Because of the different levels of learning, you need to keep in mind that nothing is obvious and at the same time, give the user the benefit of the doubt. Sometimes there is a fine line between the two, but as you spend time with your users, you will begin to know them and what particular method of support works best for them.

3 Style Files and Macros

What makes supporting TEX and LATEX even possible is the fact that so many people have contributed thousands of style files, macros, printer drivers, previewers, etc. One person can not begin to solve every problem without the help of others who have developed expertise in particular areas and who are willing to share their solutions.

My concept of support is based heavily on relying on others to solve common problems. I don't believe in re-inventing the wheel! I would rather spend the time looking around the network searching for a solution to a problem, than immediately trying to write a style file or macro to do the job. In most cases, I find the solution to my problem and a few additional helpful hints along the way.

In order to take advantage of these resources, you need to be familiar with and have access to network facilities. I feel this is an essential asset for a support person. At least you can provide a link to the outside world for your users. I'm sure, in part, the reason we have such a good support structure at the University of Delaware for TEX and LATEX is based on our ability to access the network. We can inquire and receive an immediate response. This would not necessarily be the case for those who do not have access to the network. However, at least having access to this information is better than re-inventing the wheel.

I would like to mention a few network resources I have found invaluable in providing support for TEX and LATEX.

- **Archie**
  Archie provides an electronic directory for locating software, documents, and archives on the Internet and was conceived and written by Alan Emtage and Peter Deutsch of the McGill University Computing Centre. Archie maintains a database of numerous FTP sites throughout the Internet. Each site is updated approximately once a month, ensuring that listings are kept up to date. Archie also provides a large collection of descriptions of public domain software packages and documents.

  From the "Supplementary TEX Information" file:

  The Archie server contains file listings for hundreds of ftp sites. Telnet to `quiche.cs.mcgill.ca(132.206.2.3)` and type `archie` when asked to log in. Then type `help`.

Check with your local systems administrator about an Archie server nearest to you or write to `archie-l@cs.mcgill.ca` with questions about Archie itself.

- **TEX Newsgroup**
  `comp.text.tex` is a newsgroup available via the USENET network news article and utility files system. This particular newsgroup focuses on all forms of TEX and allows for interactive question and answer discussions.
  If you mail questions and/or responses to `INFO-TeX@SHSU.edu`, then they are cross posted to the `comp.text.tex` newsgroup.
- **FAQ (Frequently Asked Questions)**
  Posted monthly by Bobby Bodenheimer (`bobby@dry.mu.caltech.edu`) is the "Frequently Asked Questions" about TEX. This file currently contains answers to 34 common questions asked about TEX and LATEX. Following is the question and answer to obtaining a copy of this information.

  How can I get a copy of this article?
  You're reading it aren't you? SAVE it :-). Seriously, though, this article is posted monthly to `comp.text.tex` and cross-posted to `news.answers`. It is therefore archived at any site that archives `news.answers`. `news.answers` is archived on `pit-manager.mit.edu` (`18.72.1.58`), and this article is available there via anonymous ftp in the directory `./pub/usenet/news.answers/tex-faq`. If you do not have anonymous ftp, send an e-mail message containing the lines `SENDME FAQ`. to `fileserv@shsu.edu` (`fileserv@shsu.bitnet`). Another way to retrieve it via email is through the mailserver at `pit-manager`: send a message containing the lines `help` and `index` to `mail-server@pit-manager.mit.edu` for information on how to obtain it.

  Other `news.answers/FAQ` **archives are:** `cnam.cnam.fr` (`192.33.159.6`) in the anonymous ftp directory `/pub/FAQ`; `ftp.uu.net` (`137.39.1.2` or `192.48.96.2`) in the anonymous ftp directory `/pub/usenet` (also available via mail server requests to `netlib@uunet.uu.net`, or via uunet's 1-900 anonymous UUCP phone number); and `archive.cs.ruu.nl` (`131.211.80.5`) in the anonymous ftp directory `NEWS.ANSWERS` (also accessible via mail server requests to `mail-server@cs.ruu.nl`). Many of the archives mentioned in question 22 also maintain current versions of this article.
- **FTP Sites, Major Archive Sites, Mail Servers, and Mailing Lists**
  Posted monthly by Guoying Chen (`chenguo@lab.ultra.nyu.edu`) is the "Supplementary TEX Information" file. This is the so-called "Supplement to the Frequently Asked Questions" file; it is intended to complement Bobby Bodenheimer's "Frequently Asked Questions" file. Note that there is some duplication of material.
  This file is monthly (in early days each month) posted to USENET newsgroup `comp.text.tex` and cross-posted to `news.answers`. This file is in several parts. It is reposited in `CS.NYU.EDU` (`128.122.140.24`) at `~ftp/pub/tex/` and also in many good archivers, e.g. `rusinfo.rus.uni-stuttgart.de` at the `/soft/tex/documentation/` as the file `FAQ.supplement-mmm.yy` (`mmm-yy` is the month and year). In that directory is also the `comp.text.tex` FAQ.

- **TEX Macro Index**
  Recently released by David M. Jones, is the first edition of the TEX Macro Index. It is available at six different anonymous ftp/or mail server sites.
  A brief description of the TEX Macro Index follows.
  This is an index of TEX macros. Its scope includes all macros that are available via anonymous ftp or mail-server or some similar mechanism. Commercial packages will be included only if a full Index entry is supplied to me by the vendor.
  Since the Index is devoted to macros, fonts and special-purpose programs are mentioned only when they are necessary to explain the purpose of a set of macros.
  Each entry is divided into several fields with the following functions:
  Name: The name of the macro package.
  Description: A short (usually 1-3 line) description of the package.
  Keywords: A list of keywords to facilitate searching for special-purpose macros, as well as to help describe the macros. A glossary of keywords can be found at the end of the file.
  Archives: A list of archives where the package can be found. Whenever known, the primary distribution site is marked with an asterisk.
  Author: The name and address (preferably electronic) of the author of the package.
  Latest Version: The date and/or version number of the latest release of the package.
  Supported: Whether or not the package is supported, that is, whether the author wants to receive bug reports and/or comments on the package.
  See also: A list of other packages with similar features.
  Note: Any additional information that seems pertinent.
  In addition to the list of packages, the Index

also contains a brief list of TEX Archives with descriptions of the services they offer.

How to Retrieve the TEX Macro Index?

1. `archive.cs.ruu.nl` (Netherlands)
   How to get `TeX-index.Z` from the archive at Dept. of Computer Science, Utrecht University:
   NOTE: In the following I have assumed your mail address is `john@highbrow.edu`.
   Of course you must substitute your own address for this. This should be a valid internet or uucp address. For bitnet users `name@host.BITNET` usually works.
   by FTP:
   (please restrict access to weekends or evening/night (i.e. between about 20.00 and 0900 UTC)).
   ftp `archive.cs.ruu.nl` (`131.211.80.5`)
   user name: anonymous or ftp password: your own email address (e.g. `john@highbrow.edu`) Don't forget to set binary mode if the file is a `tar/arc/zoo` archive, compressed or in any other way contains binary data. get `TEX/DOC/TeX-index.Z`
   by mail-server:
   send the following message to
   `mail-server@cs.ruu.nl`
   (or: `uunet!mcsun!hp4nl!ruuinf!mail-server`):
   `begin path john@highbrow.edu`
   (PLEASE SUBSTITUTE *YOUR* ADDRESS)
   `send TEX/DOC/TeX-index.Z end`
   NOTE: *** PLEASE USE VALID INTERNET ADDRESSES IF POSSIBLE. DO NOT USE ADDRESSES WITH ! and @ MIXED !!!! BITNETTERS USE USER@HOST.BITNET ***
   The path command can be deleted if we receive a valid address in your message. If this is the first time you use our mail server, we suggest you first issue the request:
   `send HELP`
2. `ftp.th-darmstadt.de` (Germany)
   The TEX Macro Index is available via anonymous ftp from
   `ftp.th-darmstadt.de`(`130.83.55.75`)
   directory `pub/tex/documentation`
   file `styles-and-macros.Index.Z`
3. `ftp.math.utah.edu` (USA)
   The TEX Macro Index is available via anonymous ftp from `ftp.math.utah.edu` (`128.110.198.2`) in the file `pub/tex/tex-index`. To retrieve it by e-mail server, send a message to `tuglib@math.utah.edu` with the subject or body
   "`send tex-index from tex`".
4. `Niord.SHSU.edu` (USA)
   To retrieve the TEX Macro Index in 8 parts suitable for electronic mail handling, include the command: `SENDME TEX-INDEX` in the body of a mail message to `FILESERV@SHSU.BITNET` (`FILESERV@SHSU.edu`).
5. `TeX.ac.uk` (UK)
   The TEX Macro Index is available via anonymous ftp, JANET NIFTP and mail server from the UK TEX Archive, `TeX.ac.uk` (`134.151.40.18`) in the file `[tex-archive.doc]TeX-index.txt`.
   To retrieve it by mail server, send a message to `TeXserver@tex.ac.uk` containing the following line
   `FILES [tex-archive.doc]TeX-index.txt`
6. `theory.lcs.mit.edu` (USA)
   The TEX Macro Index is available via anonymous ftp and mail server from `theory.lcs.mit.edu` (`18.52.0.92`) in the file TeX-index in the directory `pub/tex`.
   To retrieve it by mail server, send a message to `archive-server@theory.lcs.mit.edu` containing the following line
   `send tex TeX-index`
   The Index is also available in compressed, zip'ed and zoo'ed format in the files `TeX-index.Z`, `TeX-index.zip` and `TeX-index.zoo`, respectively. Note that if you want to request one of the compressed files by mail server, you'll have to specify a method of ASCII encoding by including one of the following lines in your mail message:
   `encoder btoa`
   `encoder uuencode`
   `encoder rscs`

In early 1991, I hoped that some type of document would be developed to provide information about macros, tools, etc. I'm glad to see that something has so emerged quickly onto the scene. I'm sure the entire TEX community will benefit from the time David has put into this Index. Many thanks David.

4 Tools

In order to support any package efficiently, you need to be able to specify a working environment that is acceptable for TEX and LATEX. For example, if you do a lot of telephone consulting, it can very time consuming and frustrating if someone calls and says "I have this error, Missing }, but I can't seem to find anything wrong." Sometimes you just want to say, "SO, what do you want me to do?" All joking aside, much of this frustration can be eliminated if the proper tools are available.

First of all, I would say that having the appropriate documentation is essential. As the support person, you should have a copy of as many of the popular TEX, LATEX, etc. manuals as possible. As a user, you should have a copy of the appropriate manual for TEX or LATEX (whichever package you would use). This avoids the

translation of information over the phone, network, etc. You can specifically refer to page numbers for solutions and hopefully the user will realize that he or she had the answer to his or her problem all along in the manual and will use it the next time before they call upon you.

Following is a list of common books that I see floating around in the TEX user community.

- *The TEXbook*, Donald E. Knuth
- *TEX for the Impatient*, Paul W. Abrahams
- *A Beginner's Book of TEX*, Raymond Seroul and Silvio Levy
- *The Joy of TEX*, Michael Spivak
- *LATEX: A Document Preparation System*, Leslie Lamport
- *LATEX for Engineers & Scientists*, David J. Buerger
- *LATEX for Everyone*, Jane Hahn
- *LAMS-TEX The Synthesis*, Michael Spivak

Secondly, I would say that there are many tools available to make using TEX and LATEX much easier. Sometimes, knowing all these tools can be difficult especially when you are supporting several computer platforms. For example, some tools are available in UNIX and not for PC's and Mac's, and vice versa. However, there are a number of tools that go across platforms and certainly are worth implementing as part of a basic configuration [5].

- Check Matching
  `texmatch` is a program that checks for matching delimiters in TEX and LATEX documents. I know this is available on most computer platforms; however, I am not sure about the Macintosh.
- Spell Checking
  `detex` is a filter that strips TEX and LATEX commands from a document. This is needed for some computer platforms before using a spell checker program.
  On the PC and Macintosh platforms, spell checkers are available that allow the user to set up a dictionary of words (which can contain control sequences) related to TEX and LATEX. This has the added value of reducing TEX errors because it is less likely to have a misspelled control sequence.
- Previewing
  Any previewer is fine as long as there is at least one available to preview a `.dvi` file. One may also consider having a previewer for PostScript, too. I have found this invaluable with trying to include PostScript into TEX and LATEX documents. In general, having a previewer is a must. This should be true not only from the user's point of view, but especially the support person's point of view.
- Including Graphics
  If you have a method that works for you, GREAT! However, if you have not committed, then consider using PostScript. Of course this means you need to get a PostScript capable printer, a DVI→PostScript printer driver, and the software you are currently using to generate your graphics must provide an option to create a PostScript file.
  The DVI→PostScript printer driver of choice is `dvips` (written by Thomas Rokicki). It is well supported and has given the best and most consistent results when trying to include PostScript graphics from many different applications. `dvips` has been ported to many different computer platforms. It provides a set of macros, `epsf`, and also works well with the `psfig` macros, where either can be used to facilitate the inclusion of PostScript into TEX and LATEX documents.
  Providing support for including PostScript into TEX and LATEX documents will become essential in the future. Over the past two years, I have seen its popularity grow and, as a result, the support I was providing needed to grow, too [3].
  1. Learn enough PostScript to be able to interpret errors and modify an existing PostScript file.
     PostScript Language Reference Manual, Adobe Systems Incorporated
     PostScript Language Tutorial and Cookbook, Adobe Systems Incorporated
  2. Learn the two macro packages available for including PostScript, `epsf` and `psfig`.
  3. Learn as much as possible about what macros or programs are available to accommodate what users might want to do with PostScript. For example, rotating, two-up, etc.
  4. Get to know the applications and printers that users will be using to generate PostScript files and run a lot of tests.

5 Summary

Although providing good support for a package like TEX or LATEX can be difficult, it is possible! It will take time and work to get to know your users, address their needs, and let them know what your expectations are as the support person.

There are three basics points to keep in mind:

1. Remember the different stages of learning as a user and support person. Try not to overestimate or underestimate your users.
2. Know what is available (or where to find existing solutions) to solve common and difficult problems. Don't re-invent the wheel!
3. Keep up with the technology.

All of these ideas are essential in providing accurate, efficient, and timely support. Hopefully, in return, you will develop a strong support network in your organization.

References

[1] CNS USER SERVICES, *The LATEX UDThesis Format.* University of Delaware, Newark, June 1990.

[2] CNS USER SERVICES, *The TEX UDThesis Format.* University of Delaware, Newark, June 1990.

[3] CNS USER SERVICES, *Getting PostScript into TEX and LATEX Documents.* University of Delaware, Newark, June 1992.

[4] ANITA Z. HOOVER, "Using WordPerfect 5.0 to Create TEX and LATEX Documents", *TUGboat* 10(4), December 1989, pp. 549–558.

[5] ANITA Z. HOOVER, "LATEX/TEX User: A Typist, or Typesetter?", *TUGboat* 12(3), December 1991, pp. 397–400.

[6] DONALD E. KNUTH, *The TEXbook. Computers and Typesetting,* Vol. A., Addison-Wesley, Reading, Mass., 1986.

[7] ROBIN L. KUBEK, "TEX for the Word Processing Operator", *TUGboat* 10(4), December 1989, pp. 561–566.

[8] LESLIE LAMPORT, *LATEX: A Document Preparation System.* Addison-Wesley, Reading, Mass., 1986.

[9] OFFICE OF GRADUATE STUDIES, *Thesis Manual: A Degree of Excellence.* University of Delaware, Newark, June 1990.

The Pursuit of Quality*

How can automated typesetting achieve the highest standards of craft typography?

Frank Mittelbach† and Chris Rowley‡

November, 1991

Abstract

This paper compares high-quality craft typography with the state of the art in automated typesetting. The first part discusses several typographical conventions which cannot be implemented by means of any formatting model currently in use. The second part explains why the current paradigms of computerized typesetting will not serve for high-quality formatting and suggests directions for the further research necessary to improve the quality of computer generated layout.

Keywords: craft typography, automated typesetting, document formatting model, paradigm, typographic rules, visual context, logical context, global optimization.

1 Introduction

The preparation of quality typeset documents requires highly skilled understanding and application of concepts developed over hundreds of years in craft typography [16]. As the constraints imposed by a particular document (i.e., its unchangeable textual and logical content) usually result in conflicts between the generally accepted rules of quality typography, some of these rules have to be violated in favour of others. The skill of the compositor lies in the resolution of such conflicts between ideals. This results in the transformation of an abstract document into a visual form which is both aesthetically pleasing and faithful to the intentions of the author.

The nature of the author's intentions, and the details of how best to realize them visually, are largely outside the scope of this paper. For us, it is sufficient to know that the placement of objects in a physical representation of some particular document is (or should be) the result of a set of rules together with some mechanism for resolving conflicts. In craft typography these conflicts are typically resolved by interaction between the typographic designer and the compositor [32].

The equivalent process for automated typesetting is a computer program whose input is a form of the document containing only the text and logical mark-up and whose output is a complete description of the detailed content of each page of the typeset form of the document. As RICHARD SOUTHALL [27] has written:

> In formatting a document on a computer-based system, the graphic characteristics of blocks of text and their arrangement on the printed page are dictated by presentation rules which are determined by the format designer. The detailed arrangement of characters and spaces within the lines of text that make up a block is also governed by rules, ...

To have any chance of emulating the traditions of craft typography a program must, in some sense, understand such rules and be able to implement them. Moreover, it must contain mechanisms for resolving conflicts between these rules. It is also important to note that the set of rules which form the basic description of the layout for a document, or a class of documents, (i.e., the concepts for positioning and shaping the contents) is potentially very large.

We have therefore split the requirements specification of a program for the production of high quality documents into two major components:

- Any of the possible rules which form the basic layout must be implementable and applicable by the program.

† EDS, Electronic Data Systems (Deutschland) GmbH, Eisenstraße 56 (N15), D-6090 Rüsselsheim, Federal Republic of Germany, Internet: mittelbach@mzdmza.zdv.uni-mainz.de

‡ Open University, Walton Hall, Milton Keynes, United Kingdom, Janet: ca_rowley@vax.acs.open.ac.uk

- Whenever the application of the rules produces a conflict, a resolving mechanism needs to be definable.

The challenge which this paper poses is, in computer science terms, to find suitable data-types whose transformations model well both the rules of the craft and the heuristics for conflict resolution. A necessary prerequisite for this is a suitable codification of the rules of craft typography and an evaluation of methods for automating its heuristics. We are not suggesting that the software should mimic exactly the skills of a craft typographer but that it should aim to produce typographical results which are, to the trained eye, indistinguishable from what would be expected from a reputable compositor.

Furthermore, we do not intend to give the impression that there is no need for well-designed systems which support interaction between a human operator and the typesetting software. However, in this paper we wish to focus on the requirements of the *automated* part of any system which aims at typographic excellence.

We start by looking briefly at a few examples of the typographer's craft which are addressed imperfectly, if at all, by existing typesetting software. Section 3 then analyses some of the basic requirements of a model for the design of software to automate craft typography and Section 4 summarizes the current state of the art; finally we suggest some directions for future research.

The authors wish to thank Richard Southall for many helpful discussions on subjects related to this work and Reinhard Wonneberger for his comments on early drafts of this paper.

2 A brief look at some rules

The rules of craft typography were developed to help the reader to understand the contents of a document [25, 26, 28, 29, 31]: good layout does not distract the reader's attention from the main aspect of the document—its message. Good typography therefore is a silent art; not its presence but rather its absence is noticeable.

Naturally, few typographical rules are universal: they depend on the purpose of the document, the cultural background of the prospective readers and many other things. Examples of the cultural aspect of typography range from obvious differences such as the direction of the typesetting (left-right, right-left or vertical) to more subtle distinctions such as the use of a type-face suited to the language of the document. Quite noticeable changes in the grey-values of paragraphs occur when the same type-face is applied to different languages ([25, p.43]), e.g., German with many capitals, French with its accents, etc.

The rules and concepts also depend on the school of typography and on particular house styles [5, 21, 6, 11]. Therefore, the discussion in this section is intended only to give some examples of rules which a designer *might* want to specify for particular documents and is not meant to be a complete survey of the subject; it focuses on rules accepted by most western schools of typography but which are not, to our knowledge, achievable by the currently available systems for automated typesetting.

2.1 Line breaking

Line breaking and its concomitant, word-division, provide a good example of the need for balance between conflicting ideals. The breaking of paragraphs into individual lines is not done simply because of the constraints imposed by a given page width—it has many aspects, including the following.

- The choice of page-size, layout, text-measure and type-face appropriate to the subject matter: long eye movements should be avoid as they tire the reader and impede the reading process.
- The quality and quantity of hyphenated lines: hyphenated words, especially when they appear too often or use psychologically bad breaks, make reading difficult (see Section 2.1.1).
- The shape of the paragraphs and the distribution of white-space within them: getting this right often leads to conflicts with the previous constraint. The use of ragged-right typesetting can help this conflict but leads to other problems, which are discussed in 2.1.2 below.

2.1.1 Word-division

Hyphenation of words is a process deeply entangled with paragraph construction [17]. All modern computerized typesetting systems incorporate some automated system for hyphenating words when the need arises, but most treat all the possible hyphenation-points as equally acceptable. As RONALD MCINTOSH suggests in [18] (and has implemented in the *Hyphenologist* software [8]), this is incorrect as they usually vary in their 'goodness' and these differences will influence the overall quality of the document. A special case of this aspect of hyphenation is that an inserted hyphen should ideally be distinguishable from a hyphen that is part of the word (e.g., re-cover vs. recover), allowing the designer to set up rules which take into account whether such a word should be broken at such a point [7].

Additionally, the distribution of hyphens within a paragraph is normally subject to typographical rules such as "avoid more than three hyphens on consecutive lines".

2.1.2 Paragraph shape and white space

When words have to be shaped into some template, two basically different approaches are common: ragged-right or justified blocks of text [30].

With justified text the intention is to give a uniform appearance: the excess space on individual lines

needs to be distributed as evenly as possible over the whole paragraph, which makes a global optimization algorithm for determining the break-points necessary [1, 22, 23]. To ensure a similar grey value over the whole paragraph, differences between the excess space in adjacent lines must also be taken into account. However, it is not only the *average* distribution of white space which is important: it is, for example, also essential to avoid 'rivers' of white space flowing vertically through a paragraph [31].

Another refinement is that white space between words needs to be different for different combinations of boundary characters in order to give a uniform appearance [19].

If, on the other hand, ragged-right shapes are desired then the typesetting system must be capable of producing aesthetically pleasing shapes [26, p.131], e.g., avoiding successive lines of equal length and extremely uneven shapes (if the layout description specifies this). The absence of literature on this problem and the failure of any current typesetting systems to implement such rules suggests that the underlying concepts of ragged-right typesetting are not at all well understood.

Most of the problems covered here can, with most currently available systems, be detected only by human eyes. Furthermore, their circumvention is generally difficult and, at best, involves a large amount of detailed ad hoc interaction with a skilled human operator. That they have not at present been automated is in large part because the concepts necessary to describe and evaluate them have not yet been defined.

We are aware that there exist automated systems which go some way towards tackling particular aspects of the production of visually optimal paragraphs. The methods they use include the following: assessment of the 'average grey value'; use of 'weighted word-division points'; control of multiple consecutive hyphenations; optically correct inter-word spacing and optical alignment of the margins. However, we know none which do all of these, and none which make any explicit attempt to avoid 'rivers'

2.2 Grid layout

The eye needs a regular grid of points on which to focus when searching for objects (e.g., the next line) on a page. Many designers therefore base all aspects of the page-layout of a document on an underlying grid [13]. Such layout constraints are fundamental to the typesetting paradigms of many of the widely used DTP software packages but, regrettably, this is the *only* concept relevant to high quality typesetting which is understood by most such systems!

The distribution of white space in such a layout is necessarily discrete, which throws optimizing systems based on the box-glue model [14, 12] out of balance.

Since such a layout limits variation in the placement of constructed objects (like broken paragraphs, constructed figures with captions, etc.), to achieve good quality within such constraints there needs to be a high level of interaction between the building mechanisms for these objects and the page make-up mechanism (or, more generally, the algorithms for constructing higher level units). Such interaction is not implemented in any current systems (Section 3 contains a more detailed discussion of this topic).

2.3 Placing floating objects

Floating objects (e.g., footnotes or figures) are linked to the main body of a document by means of cross-references or other visual clues intended to identify them for the reader. Rules for their positioning can become quite complex to implement, even when they are easy to state [21]: e.g., that the distance between the reference and the float should ideally be small, or that the float should be visible from the point of its major reference.

Since floats should be easily distinguishable by the reader as individual units, clear visual separation from the main body of the text is usually necessary. This is typically achieved by font changes, extra space, rules, etc. thus such objects can quite drastically alter the visual appearance of a page. For this reason some designers prefer, for example, to lay out all the figures on a double-page spread as a single visual unit (placing individual figures and captions according to their size and form) to produce a balanced look.

To allow for such designs the software needs to be driven by heuristic rules as opposed to the procedural routines used in all current systems. Again, to emulate craft typography it is essential that complete interaction between this part of the system and all the other formatting routines is maintained: for example, it is necessary to allow paragraphs which flow around arbitrary figure shapes to, nevertheless, have globally optimized line-breaks.

Again, no currently available software implements more than a minimal amount of automation in this area.

3 Choosing the context

Each of the rules from the last section (along with many others) poses an intrinsically demanding and interesting implementation problem worthy of research effort—but it is not only the specification of certain important typographical conventions which is difficult or impossible within current systems. There is a more fundamental limitation which will prevent these systems from approaching craft standards, however good their formatting mechanisms may be—they have all been developed under the following paradigm:

> *That the formatting parameters for any object in some document can be determined by sequentially parsing its logical form.*

In other words, that these parameter values can be predetermined in such a way that the document can be processed in one pass (except for things like forward cross-references, which are resolved in a subsequent pass).

We claim that this approach will never suffice for the high quality standards of craft typography. As RICHARD SOUTHALL [27] went on to point out:

> With complex text, the achievement of character arrangements in the printed output that help to clarify its meaning to the reader may require differing rules of composition to be used in the formatting of different parts of the text.

We would extend his observation by pointing out that each object in a document should be formatted according to the context in which it finally happens to fall. This context has two components: its logical context and its visual context.

By **logical context** we understand the placement of the object (i.e., of its tag in the logical mark-up of the document) with respect to the other objects of the document. This component of the context is, in principle, fixed for a particular document (e.g., in a simple model for the logical structure of a document, it is given by the nesting and sequencing of the logical tags) but its analysis may involve substantial look-ahead or even a preliminary pass over the whole document.

The **visual context** of an object consists of the visual concepts and rules which are active at the place in the formatted document where the object actually appears (this includes, for example, the page(s) on which it falls, whereabouts on a page it appears and what else is on nearby pages). This visual component usually cannot be determined without complete knowledge about the placement (and consequentially the formatting) of all the other objects in a particular document. For example, the placement and measure for a figure caption might depend on whether that figure falls on a verso or on a recto page or whether there are other figures present on this particular page. Thus, if a 'floating figure' floats from a page of mixed text/figures to an 'all figures' page, its logical context will not change (since this is determined by the first reference to it in the text) but its visual context may change significantly.

Whilst the correct logical context for any object can in principle be determined after a single scan through the whole document, this is certainly not possible for its visual context since this depends on the formatting of all other objects within the constraints given by the applicable typographic rules. The above paradigm must therefore be replaced by one which recognizes that:

- formatting a document is a global process which must take into account variation in the visual context of each object;
- an iterative process is required to achieve an optimal (or even a high-quality) result.

Our basic model of automated document formatting is thus very similar to the model described in Section 1.2 of [10]. However, our refinement of the model is radically different from that developed in Section 1.3 of that article: this is probably an accurate reflection of the fundamental tension between the needs of a WYSIWYG system and those of automated high-quality typography. To achieve craft quality, detailed decisions must be made concerning the typographical treatment of all aspects of a document (from the layout of a spread down to the individual character glyphs). Such decisions depend, for a particular object, not only on its logical position in the abstract document but also on its visual relationship to the rest of the formatted document.

4 The current situation

The only major project which has addressed the problems of automating high quality typesetting remains that of Knuth (and developments thereof). There seems to have been nothing published on the theoretical side of this subject since 1982. The progress to that date is well described in *Document Preparation Systems* [20] from which we would particularly recommend the comprehensive survey by RICHARD FURUTA et al [9]. In this article the distinction between the editing process and the formatting process is clarified and a number of systems are described—both pure formatters and integrated editor/formatter systems. The authors then point out, in Section 4.5.1, that since all systems leave much of the task of producing satisfactory formatting to the user, close integration of the editing and formatting is essential since this makes "the generation of the concrete document part of a single document creation process".

Perhaps this explains the rapid advances over the last decade in the development of integrated editor/formatters, resulting in the present-day sophistication and range of systems incorporating the WYSIWYG paradigm of document formatting. By contrast, there has been very little progress in automating the formatting process itself and thus rendering unnecessary the human time and skills needed to produce high quality work within the WYSIWYG paradigm. It is encouraging to note that the emerging DSSSL standard [2] does allow expression of our paradigm—but this establishes no more than a common language in which to express these ideas.

In the TEX system the concept of global formatting, allowing for variation of visual context, is realized up to a certain point. In particular, the paragraph builder implements our paradigm to a limited extent: it evaluates a large number of variant possibilities for the visual context of the objects (characters and inter-word spaces) and globally (up to the paragraph boundary) optimizes

over them within a quite complex parameter space. For example, even if a character sequence (such as `ff`) would be typeset as a single ligature glyph in the visual context of a line of type, when it is broken (by hyphenation) across a line-boundary the visual context of the sequence is changed so that two separate glyphs are typeset. TEX's paragraph builder has recently been extended by allowing, when no sufficiently good solution is found, a 'rescaling' so that a larger region of the parameter space is searched for an optimal solution [15].

Following on from the work of Knuth, but still working under the assumption that the visual context, and thus the formatting parameters, of certain entities could be predetermined, a globally optimizing pagination algorithm was analysed and implemented by MICHAEL PLASS in his thesis [24] and such an algorithm has been incorporated in the Type & Set System developed by GRAHAM ASHER [3, 4]. Both use a two step procedure in which the first run is used to produce a galley form of the document. In this form, textual items are already composed into lines, figures, etc. and their formatting is not therefore subject to any further refinement. In the second phase these preformatted objects are used to construct the final pages. This phase is controlled by an optimizing algorithm which takes into account the "distances" from references to figures together with "grey-values" (glue-stretching) for individual pages.

Even though these systems implement considerably more quality-oriented features than most other systems, they are still severely limited in many ways: e.g., they have no explicit constructs, or rules, designed to detect and prevent 'rivers'. Also, they do not conform to our paradigm in other important areas. For example, the page-building mechanism receives paragraphs already irrevocably broken into lines with no possibility of requesting a retry to find a variant which improves the page-breaking. The mechanism used by TEX itself to discourage a hyphen at the end of the last line on a page is simply to penalize a possible page-break after this line: it does not recompose the paragraph to avoid the problem hyphen by use of a slightly different sequence of line-breaks.[1]

5 New directions

The problems involved in the automation of typesetting are numerous and none of the currently available computer programs addresses more than a small number of them. As a result, documents produced by present-day software (without a significant amount of human interaction) do not come anywhere near the standards of craft typography. (The use of TEX in conjunction with the very simplest of typographic designs is perhaps an exception to this statement.)

The suggestions we make in this section for future research are directed mainly at computer scientists. However, this does not mean that we believe that they can solve the problems by themselves—on the contrary, much collaborative work with designers and compositors is essential to this task.

Implementing in full the concept of global formatting either means optimizing the formatting of the document as a whole (i.e., storing information about *all* possible variants in the formatting of every entity) or it means emulating this process by means of an iterative process—the latter appears to be the only practical option.

In all currently available systems, formatting is implemented by procedural routines based on bottom-up concepts, i.e., larger units are constructed from smaller ones whose internal format is already fixed. To achieve higher quality in automated typesetting, the optimization part of the process cannot be confined in this way to individual layers of complexity, e.g., first all paragraphs, then all pages (as in Type & Set). Instead, each individual routine which formats a particular object needs to produce as its output a range of variant formattings (each locally optimized) for that object under one or more assumed context possibilities. For example, a paragraph formatting routine might return information such as "this paragraph could be laid out in the given context into 5 lines with a forbidden break after line 2, or into 6 lines with . . . ".

The information thus gathered can then be the input to a global optimization process which produces a new formatted view of the document with, in general, altered visual contexts for certain entities. This process should then be iterated until a fixed-point is reached, i.e., until successive iterations give the same mapping of entity and context pairs.

The specification of layout rules such as those discussed in the previous sections by means of a large collection of parameters is certainly not intuitive and requires a thorough understanding of the underlying algorithms in order to predict the results of even small changes to the values of the parameters. Moreover, this approach is defective because the only rules which are specifiable are those whose underlying concepts are present in the model and algorithms used.

In view of the nature and quantity of these rules, and hence the computational complexity of the parametric optimization problems to which they lead, it would seem sensible to investigate whether other areas of computer science can be applied to this task. In particular, we suggest the following questions.

- Can certain aspects of the rules and heuristics be effectively modelled using an expert-system approach?

[1] It is interesting to note that the TEX model for optimizing the line-breaks within a paragraph is flawed in this case since it erroneously takes into account the visual incompatibility of two neighbouring lines which end up separated by a page-break!

- Can conventional optimization algorithms be replaced by algorithms which will, with a high probability, quickly produce a near-optimal solution?

A necessary precursor to any such investigations must be the study of new conceptual models of document formatting together with accompanying description languages designed to capture the rules and heuristics in a natural manner whilst being precise enough to drive the formatting software.

We are working on the formulation and refinement of such a model of document formatting and we are studying the data-types and transformations required to implement it. However, much work needs to be done in collaboration with typographers and typographic designers in order to adequately understand the rules and heuristics of the craft which need to be built onto our model and its description language.

References

[1] James O. Achugbue. On the line breaking problem in text formatting. *Proc. of the ACM SIGPLAN/SIGOA*, 2(1,2), 1981.

[2] S. Adler, project editor. *ISO/IEC CD 10179: Information technology—Text and office systems—Document Style Semantics and Specification Language*. American National Standards Institute, New York, 1991.

[3] Graham Asher. Type & Set: TEX as the engine of a Friendly Publishing System. In Malcolm Clark, editor, *TEX applications, uses, methods*, pages 91–100. Ellis Horwood, Chichester, 1990.

[4] Graham Asher. Inside Type & Set. *TUGboat*, 13(1), (to appear).

[5] Judith Butcher. *Copy editing: the Cambridge handbook*. Cambridge University Press, Cambridge, second edition, 1981.

[6] *The Chicago Manual of Style: Rules for authors, Printers and Publishers*. University of Chicago Press, Chicago, 13th edition, 1982.

[7] Carl Dair. University of Toronto Press, Toronto, 1967. Paperback reprint 1985.

[8] David Fawthrop. *Hyphenation by algorithm of English/American and other languages*. Computer Hyphenation, Bradford, 1990.

[9] Richard Furuta, Jeffrey Scofield and Alan Shaw. Document Formatting Systems: Survey, Concepts and Issues. In [20].

[10] Bo Stig Hansen. A function-based formatting model. *Electronic Publishing*, 3(1):3–28, 1990.

[11] Sue Heinemann and Virginia Croft, editors. *Xerox Publishing Standards: A Manual of Style and Design*. Watson-Guptill, New York, 1988.

[12] Mary Anne Holstege. *Marking and the Design of Notations*. PhD thesis, Stanford University, Department of Computer Science, Stanford, CA 94305, June 1989. Report No. STAN-CS-89-1270.

[13] Allen Hurlburt. *The grid: A modular system for the design and production of newspapers, magazines, and books*. Van Nostrand Reinhold, New York, 1978.

[14] Donald E. Knuth. *The TEXbook*, volume A of *Computers & Typesetting*. Addison-Wesley, Reading, Massachusetts, May 1989. Eighth printing.

[15] Donald E. Knuth. The new versions of TEX and METAFONT. *TUGboat*, 10(3):325–328, 1989.

[16] Hans-Joachim Koppitz, editor. *Gutenberg Jahrbuch*. Gutenberg-Gesellschaft, Internationale Vereinigung für Geschichte und Gegenwart der Druckkunst e.V., Mainz. Contains results on the past and present history of the art of printing. Published since 1926.

[17] Franklin Mark Liang. *Word Hy-phen-a-tion by Com-put-er*. PhD thesis, Stanford University, Department of Computer Science, Stanford, CA 94305, August 1983. Report No. STAN-CS-83-977.

[18] Ronald McIntosh. *Hyphenation*. Computer Hyphenation, Bradford, 1990.

[19] Frank Mittelbach. E-TEX: Guidelines to future TEX extensions. In Lincoln K. Durst, editor, *1990 Conference Proceedings*, pages 337–345, September 1990. Published as TUGboat 11#3.

[20] Jurg Nievergelt, Giovanni Coray, Jean-Daniel Nicoud and Alan C. Shaw, editors. *Document Preparation Systems*. North-Holland, Amsterdam, 1982.

[21] *Hart's Rules; For Compositors and Readers at the University Press, Oxford*. Oxford University Press, London, 39th edition, 1991.

[22] Michael F. Plass and Donald E. Knuth. Breaking paragraphs into lines. *Software—Practice and Experience*, 11:1119–1184, 1981.

[23] Michael F. Plass and Donald E. Knuth. Choosing Better Line Breaks. In [20].

[24] Michael Frederick Plass. *Optimal Pagination Techniques for Automatic Typesetting Systems*. PhD thesis, Stanford University, Department of Computer Science, Stanford, CA 94305, June 1981. Report No. STAN-CS-81-970.

[25] Emil Ruder. *Typographie; Ein Gestaltungsbuch*. Niggli/Hatje, Heiden, Stuttgart, fifth revised edition, 1988. Contains complete English and French translation.

[26] Manfred Siemoneit. *Typographisches Gestalten*. Polygraph Verlag, Frankfurt am Main, second edition, 1989.

[27] Richard Southall. Presentation rules and rules of composition in the formatting of complex text. In *SGML & TEX Conference 1990, Groningen*. Abstract of talk on Page 18 of the conference program.

[28] Jan Tschichold. *Ausgewählte Aufsätze über Fragen der Gestalt des Buches*. Birkhäuser Verlag, Basel, 1987. Second printing.

[29] Jan Tschichold. *Leben und Werk des Typographen Jan Tschichold*. Saur, München; New York; London; Paris, second edition, 1988.

[30] Alex White. *How to spec type*. Watson-Guptill, New York, 1987.
[31] Jan White. *Graphic Design for the Electronic Age*. Watson-Guptill, Xerox Press, New York, 1988.
[32] Hugh Williamson. *Methods of Book Design*. Yale University Press, New Haven, London, third edition, 1983.

Writing Reports with More than a Hundred People*

Walter van der Laan and Johannes Braams

PTT Research Neher Laboratories
P.O. Box 421
2260 AK Leidschendam
The Netherlands
W.vanderLaan@research.ptt.nl; J.L.Braams@research.ptt.nl

June 1992

Abstract

This paper describes a system that produces project status reports using LaTeX. The reports contain both textual and financial information. The textual part of the status reports is written by over a hundred people who don't need to know what LaTeX is. The financial information is retrieved from a database.

1 Introduction

Each quarter of a year the clients of our research center receive status reports concerning all of their projects. As you might expect these reports are typeset using LaTeX. What might be more of a surprise is the fact that the status reports are written by over a hundred project managers who don't need to have any knowledge of LaTeX. They write their status reports at different sites, using various computer systems, word processors and editors.

The first section gives an impression of the environment and history of this reporting system. The second explains the least that a project manager needs to know when using the system. A mail server is used to collect all project status reports. This server is discussed in the third section. The fourth section describes the generation of reports. Some general conclusions are listed in the last section.

2 Environment and History

With about 95,000 employees, PTT is the largest company in the Netherlands. The business of PTT is selling postal and telecommunication services. The reporting system described in this paper was made for PTT Research, a division of PTT with about 800 employees. PTT Research does most of its work under contract to other PTT divisions; typically there about 350 research projects for about 30 PTT divisions.

Each quarter, all project managers write a one-page status report to keep their client informed. These status reports consist of the following four sections:

- a short description of the project,
- the targets for the reporting period,
- the work realized in the reporting period, and
- the targets for the next period.

For each project the text written by the project manager is pasted into a form supplied by our financial department. This form shows information from a financial package, e.g.:

- the names of the client, project and project manager,
- important dates related to the project,
- the amount of money spent so far, and
- the budget.

The completed forms are bundled and presented to our clients.

The production of these status reports used to be manual. The texts supplied by the project managers, showing all kinds of fonts and printing qualities, were literally pasted onto the form using scissors and glue. Throughout our company, about a dozen people used to be busy collecting status reports and completing forms. It took more than a month before we were able to present the status reports to our clients. During this production process it was almost impossible for a project manager to make any corrections.

The system described in this paper offers much more flexibility. People, both with and without any LaTeX experience, send their status reports to a mail server. A report generator is used to produce LaTeX files containing a mix of information from the financial database and the status reports that have been received through the mail server. It enables us to present a uniform and beautiful report to our clients about ten days after the

*Presented at TUG '92, July 27–30, Portland, Oregon,USA.

financial closure of a quarter year. Within these ten days, project managers and their managers get several chances to correct the status reports.

3 Writing Status Reports

The status reports can be written as plain ASCII text or as LaTeX text. The first option is default and foolproof, it protects a project manager from any LaTeX errors. While this option is easy to use it also leaves the writer with only a few of the expressions available in LaTeX, namely paragraphs and some special characters common in the Dutch language. This has proved to be sufficient for 99 percent of the reports but leaves a few special cases, e.g., some project managers want to put formulas in their report and others have a need for symbols used in physics. In these cases the project managers write their report as LaTeX text and they have to know how to use it. The remainder of this section explains what a project manager needs to know when using the foolproof mode, i.e. plain ASCII text.

A report sent to the mail server may contain nothing but plain ASCII characters. This is a necessary constraint because the mail server receives reports written with about twenty different kinds of word processors and editors on about five different kinds of platforms. Fortunately all word processors have an option to save a text in ASCII. This means that all project managers must know how to produce an ASCII file containing their report. They must also be aware of the limitations of ASCII; e.g., no underline, no boldface, no special characters.

Project managers must be able to send their reports to the mail server. This hasn't been a problem in our organization as everybody is connected to the local network and most people are using electronic mail.

Each report must contain a few keywords. This simple syntax is needed to

- assign a project number to each report,
- separate the four pieces of text in each report, and
- separate the report from the mail headers and footers.

The next example shows the report for project number 12345. As you can see the syntax consists of uppercase keywords with four pieces of text in between:

```
PROJECT 12345
DESCRIPTION
We are working on a project.
CURRENT TARGET
Our plan was to finish the project.
REPORT
We've had a lot of problems but the
work is almost done.
NEXT TARGET
We'll finish the project in the next
quarter year.
END
```

Many people were having problems with this combination of a textual report and a strict syntax. We eliminated this problem by extending the mail server to accept all syntax errors that occurred.

Project managers have to understand the effect of an empty line in their texts: All text is aligned on the left and right margin. An empty line causes the alignment to restart at the beginning of the next line, because texts are set without paragraph indentation.

In the Dutch language one frequently finds characters such as é, è and ï. For this reason the project managers have the option of using the sequence `backslash accent vowel` whenever they need to put an accent over a vowel.

4 The Mail Server

All mail sent to a dedicated network address is processed by a program, which replies to every message received. The reply consists of two parts. The first part contains success and error messages, e.g.: "`i found a report about project 12345`", or: "`i removed some control characters from your text`". The second part is a copy of the received message in which all recognized texts have been removed. If all went well the second part contains nothing but keywords and mail headers. This construction is clear even to people who aren't used to syntax errors.

In the beginning we had trouble with errors in the project number. People would erroneously send us a report for project 12435 instead of 12345. This meant one could overwrite another report by mistake. We solved this problem by saving a list of the mail addresses of the senders for each report. We accept reports from any address on the list. A report from any other address is rejected but the address is added to the list. In such cases the sender receives a message saying the report has been rejected but will be accepted if the report is sent again. This warning eliminates the typing errors in project numbers but still allows different people to send updates for the same project.

In the foolproof mode all texts must be transformed into valid LaTeX texts. This transformation is described next.

Many characters special to LaTeX have to be deactivated. We handle those characters as four separate cases:

- Double quote characters are replaced by " and ", respectively.
- A backslash is added in front of the following characters: #, $, %, &, _, { and }.
- The math characters <, > and | are placed between dollar signs.
- The expression `{\tt\char92}{}` is used to insert a backslash. Note that 92 is the ASCII value of a backslash. The same procedure is used to insert

the characters ^ and ~.

The sequence `backslash accent lowercase vowel` is allowed. This sequence isn't changed by the transformation; only when the vowel is an *i* is it replaced by a dotless ı. This change makes the sequence very uniform and easy to understand for people not used to LaTeX.

LaTeX does a great job at automatic hyphenation. However, when words are joined together by characters such as the slash and minus, LaTeX doesn't hypenate the resulting string. This causes a lot of overfull hboxes. To solve this the transformation program looks for the sequence `letter slash` or `minus letter`. Within sequences like this the slash or minus is transformed into the parameter of the `nw` command (nw stands for *new word*), e.g.: `man/woman` is replaced by `man\nw{/}woman`. The `nw` command is expanded to let "man" and "woman" be separate words divided by a slash. The LaTeX definition is:

```
\newcommand{\nw}[1]
  {\hspace*{0pt}#1\hspace*{0pt}}
```

This transformation has proved to be sufficient in avoiding almost all overfull hboxes.

Last but not least, all characters unknown to LaTeX are removed during the transformation.

The *Lex* specification below (Lesk and Schmidt, 1975) produces a program that performs the transformation described above:

```
AN              [a-zA-Z0-9]
AC              [''"^.=]
%%
                int dq = (0 == 1);
\"              { dq = !dq;
                  if (dq) printf ("``");
                  else    printf ("''"); }
[#$%&_{}]       { printf ("\\%s", yytext); }
[|<>]           { printf ("$%s$", yytext); }
[~\^\\]         { printf ("{\\tt\\char%d}{}",
                    yytext[0]); }
\\{AC}i         { printf ("\\%c{\\i}",
                    yytext[1]); }
\\{AC}[aeou]    { printf ("%s", yytext); }
{AN}[/-]{AN}    { printf ("%c\\nw{%c}%c",
                    yytext[0], yytext[1],
                    yytext[2]); }
\t              { putchar (' '); }
\n              { putchar ('\n'); }
.               { if ((yytext[0] >= 0x20)
                  &&  (yytext[0] < 0x80))
                    printf ("%s", yytext); }
```

Text sent in LaTeX mode is not transformed by the mail server. A copy of the text is transformed into a complete LaTeX document by adding a header and a footer. The mail server passes this document to the TeX compiler and afterwards checks the log file for errors. If an error occured, it rejects the text and adds the compiler messages to the reply.

The mail server program was written using *TPU*, the VAX/VMS Text Processing Utility (Digital VMS manuals, 1988). *TPU* and LaTeX make a great team and have allowed us to build this system in a short time. If you ever need a utility to process text, try *TPU*.

5 Report Generation

The four pieces of text per project are stored in a separate directory as four files per project. All of these texts have been tested or transformed by the mail server. These texts are ready to be typeset in any textwidth or font.

Our relational database system stores a lot of project information. This information is transformed into LaTeX strings just as the texts supplied by the project managers are transformed.

With an application program our financial department can define which projects are to be included in a report and in what sequence they are to be included. Every set of projects defined can be printed in several ways; e.g., a report containing:

- all information about a project on one page,
- only the financial information, and/or
- only the description of each project.

When we started work on automating the report generation process we first agreed upon an interface between the report generator and LaTeX that consists of a few special commands.

In Figure 1 an example of a status report is shown. At the top of the page some general information about who ordered the project and who runs it is shown. This information is repeated on a continuation page as you can see in Figure 2. Then follows a short description of the project and its targets. In this case the description is too long and is therefore continued on a second page. The next texts discuss the targets for the reporting period, the work done in the reporting period and the targets for the next reporting period. At the bottom of the page some financial information on the project is included. Both this financial information and the general information at the top of the page is extracted from the database.

For each of the text fields a LaTeX environment is defined. The task of these environments is to store the text parts in four boxes of the right size. To pass the information that is printed in the header and footer of a report we defined a few LaTeX commands with parameters. All information that belongs to a particular project should be inside yet another LaTeX environment. The task of this environment is to:

- start a new page,
- select the correct page style,
- write some information about the project to a table

of contents file, and
- make sure that all information accumulated is put on the page.

Originally, the layout of the report form was defined in terms of a number of characters per line, and a number of lines for each of the four text fields. It was also specified that a report for any project should occupy not more than one page.

With typeset text, the specification of the number of characters per line is not particularly useful, because it can vary with the kind of characters that are used in the text. Also, the manual process of putting together the status reports had shown that sometimes a project manager would produce more text than would fit in the field for which it was meant.

Because the sizes of the fields are fixed, we had to choose what to do. We could typeset the portion of the text that would fit into the field and either
1. let the rest print over the next field, or
2. discard the rest, or
3. store leftover text and print it on a second page.

Option 1 clearly is unacceptable; the result would be both ugly and unreadable. Option 2 would possibly result in texts ending in a weird manner. Choosing this option would, however, force the project managers to be brief. It was finally decided to use option 3. One of the reasons for this decision was that the implementation of options 2 and 3 is almost the same.

The implementation of this part of the LaTeX style file is based on the use of the `\vsplit` command. The four environments discussed before scoop up the text and typeset it in a `\vbox \pickup@box` of the appropriate width. The contents of the `\pickup@box` are copied into a different box for each environment using the `\vsplit` operation. The resulting height of the `\pickup@box` is then measured. If it is not zero there is more text than fits into the field. In that case an indication that there is more text is appended to the first part of the text, and the rest is stored away to be put on a second page.

6 Conclusions

The system imposes no special organization upon its users. Our users write and send their reports using the computer system that they use for their everyday work. In some departments the reports are gathered by one person who sends all reports and report updates to the mail server. In other departments all project managers send and update their own reports. The nice thing about a mail server is that it doesn't mind where a report was made or who made it.

The two modes, LaTeX or foolproof, have made the system both very flexible and very easy to use. No special training is required for people who use the system in the default foolproof mode. People with special requirements, on the other hand, are very happy with the LaTeX mode.

LaTeX separates the contents of a document from the form in which it is presented. This separation was of great benefit to us during the development of the system. It allowed us to make a very clean and easy division of the work to be done. After defining the different styles needed for our system one of us would work on the generation of LaTeX reports using the defined styles, while the other would work on the creation of the styles.

At first we implemented the reports to be generated interactively. This didn't work very well because LaTeX consumes large amounts of cpu time. At the moment the production of reports is implemented as a lower priority background process.

References

[1] Lesk, M.E., E. Schmidt. "Lex — A Lexical Analyser Generator." Bell Laboratories. Murray Hill, New Jersey, October 1975.

[2] *VAX Text Processing Utility Manual.* Digital VMS Manuals, **5B**, April 1988.

PTT Research Kwartaal Rapportage

92/01

Projectnaam	PART. WERKZAAMHEDEN SC18	Projectnummer	61006
Opdrachtgever	RAAD VAN BESTUUR RVB	Onderdeel/hafd	TI INFORMATIEDIENSTEN
Afdeling	RVB	Hoofd	TERPSTRA, A.P.
Hoofd	J. WAGE (secretaris LBR)	Projectleider	W. REMMERS
Contactpersoon	W.J. THIEME	Begin/Eind datum	10192 / 311292
Kenmerk	OVK D.D. 15/07/1990	Kenmerk offerte	91-A041

Korte omschrijving en doel van het project

Binnen ISO/IEC JTC1 is SC 18 gericht op de standaardisatie-aspecten van creëren en manipuleren van documenten. SC18 omvat de volgende gebieden:
* document architecturen en semantiek (ODA, SGML, DSSSL)
* document interchange formaten (ODIF, ODL, SDIF, DTAM)
* basis inhoudsarchitecturen (o.a. teken, raster, muziek) en weergave-aspecten (kleur, fonts)
* hypermedia (HyTime, SMDL)
* printeraansturing (SPDL, DPA)
* message handling en message store (X.400 of MOTIS)
* document filing & retrieval (DFR)
* User System Interfacing (keyboard layouts, Icons & Symbols, dialoog standaarden)

Zie volgende pagina

Gesteld doel voor dit kwartaal

Gezien het belang van de SC18-onderwerpen, de breedte van het aandachtsgebied en de doelstelling de werkmaatschappijen meer bij dit werk te betrekken, dient er een duidelijk plan van aanpak geformuleerd te worden. Vanwege een wisseling in de bezetting is deze aktiviteit vorig jaar opgeschort. Hieraan zal binnenkort wel veel aandacht besteed worden.

Korte rapportage

Op dit moment zijn een groot aantal standaarden en/of uitbreidingen op de basis-standaard (zie boven) in bijna-finale vorm. Tegelijkertijd blijken er nog grote definitie-problemen te bestaan, en blijkt tussen verschillende standaarden grote overlap te bestaan. In vorige rapportages is al genoemd de relatie tussen 'character, graphic character, font en glyph', Message Store Extensions versus DFR en DTAM, ODA en SGML. tegelijkertijd worden er in hoog tempo nieuwe werk-items voorgesteld. De wens naar stabiele standaarden is momenteel strijdig met de momenteel gehanteerde ISO-planning en het aflopen van de studie-periode van CCITT. De voorjaarsvergadering is bij OC\'E gehouden, waar een interessante presentatie over de geschiedenis van het bedrijf en de bemoeienis met ODA werd gegeven.

Gesteld doel voor het volgend kwartaal

1. Het aktief volgen van ontwikkelingen binnen SC18.
2. Afronden plan van aanpak (tweede kwartaal 1992)

Inzet S&O	afgelopen kwartaal	t/m afgelopen kwartaal	overeenkomst	perc. gerealiseerd
Aantal Uren	81,00	81,00	320,00	25,30
Bedrag (fl)	16.038,00	16.038,00	63.360,00	25,30

1 Uitsluitend voor gebruik binnen PTT

Figuur 1: *A sample report for a project*

PTT Research Kwartaal Rapportage

92/01

Projectnaam	PART. WERKZAAMHEDEN SC18	Projectnummer	61006
Opdrachtgever	RAAD VAN BESTUUR RVB	Onderdeel/hafd	TI INFORMATIEDIENSTEN
Afdeling	RVB	Hoofd	TERPSTRA, A.P.
Hoofd	J. WAGE (secretaris LBR)	Projectleider	W. REMMERS
Contactpersoon	W.J. THIEME	Begin/Eind datum	10192 / 311292
Kenmerk	OVK D.D. 15/07/1990	Kenmerk offerte	91-A041

Vervolg Korte omschrijving en doel van het project

* Distributed Office Application Model (DOAM)
Vanuit verschillende optieken (EDI-wereld, kantoorautomatisering, uitgevers, enz.) worden eisen gesteld aan dergelijke systemen. De kontakten met andere standaardisatie-lichamen (CCITT, ECMA, enz.) zijn daarom intensief.

2 Uitsluitend voor gebruik binnen PTT

Figuur 2: *The second page of the sample report*

TEX-based Production at the AMS*

Ralph Youngen†

American Mathematical Society,
P.O. Box 6248,
Providence, RI 02940, USA

June 1992

1 Background

The American Mathematical Society (AMS) is a both major publisher of mathematical research, and a professional organization whose members are mathematicians engaging in research at academic institutions and other research centers in the U.S. and around the world. A primary function of the Society is to provide channels of communication whereby these mathematicians can communicate the results of their research to each other, and to the broader scientific community. Foremost among these channels of communication is an extensive publications program which is based on the TEX typesetting system.

The AMS headquarters are in Providence, Rhode Island, and a sister office affiliated with the University of Michican is located in Ann Arbor, Michigan. The two sites employ a total of more than 200 people.

Computing hardware at the AMS consists of a VAX/VMS 6320 cluster, a 2-processor Solbourne 5E/902-64 which functions as a database server, a 4-processor Sun 690 MP which is the central server for the Ann Arbor office, a 2-processor Solbourne 5/500 which functions as a TEX server for the VMS cluster, approximately 12 other Unix and VMS workstations which serve as development machines, and approximately 20 PC-compatible and 20 Macintosh computers. High resolution output devices include two Autologic APS-μ5 phototypesetters, and an Agfa/Compugraphic 9600 PostScript imagesetter. Proofing devices include two high-speed QMS 300dpi laser printers, and approximately 20 other 300dpi laser printers. A 300dpi flatbed scanner is used for input of graphic material for the creation of line art and halftones.

The Society currently publishes 20 journals, ranging from tri-weekly's to annuals, as well as nearly 100 books per year, of which 95% are in 20 regular book series. Of the journals, six contain results of primary research, eight are translations from Russian, Japanese, and other languages, and the largest, *Mathematical Reviews (MR)*, contains reviews and extensive indexes of current mathematical literature that together occupied more than 10,000 pages in 1991. AMS books and journals combine for about 70,000 pages per year, with 62,000 typeset pages and 8,000 author-prepared pages. In addition to its own publications, the AMS also does some portion or all of the composition, editorial, printing, binding, and distribution for nine other journals published by eight scientific societies. Furthermore, the AMS acts as a service bureau for typesetting jobs composed using the TEX typesetting language. Aside from a conventional desktop publishing segment for the production of a few newsletters, brochures, and posters, and any publications produced from author-supplied copy, all other publications are typeset using TEX, with the final camera copy generated on a phototypesetter at the AMS headquarters.

2 History and Importance of TEX at AMS

The AMS has used electronic publishing methods to produce its publications since 1971. Prior to the early 1970s all mathematical typesetting was done either by using typewriters or hot lead-type machines (Monotype for example). The AMS experimented with various electronic publishing systems through the early to mid 1970s, including paper-tape systems, until settling for a time with a system developed by Science Typographers Inc. (STI). The STI system was able to handle jobs containing mathematical material, but was not useful for multi-column administrative publications such as the AMS membership list or catalog.

In 1978 the AMS learned of a new typesetting system called TEX when Prof. Donald E. Knuth of Stanford University presented the Gibbs lecture at the AMS annual meeting in January 1978. Every year, a well-known mathematician is invited to present the Gibbs lecture on a topic of his or her choosing. Knuth chose to speak about mathematics and computer science in the service of technical typesetting. In the audience were several influential members and officers of the AMS, and they undertook the investigation into TEX and encouraged its adoption as a production tool for the Society's publishing program. Knuth furthered this

*Presented at the 9^e NTG meeting, June 4, 1992, Amsterdam.

†Technical Support Manager, American Mathematical Society.

adoption process by assigning the trademark rights to TEX to the AMS.

The promise of the TEX typesetting system was to provide a new typesetting system intended for the creation of beautiful books—and especially for books that contain a lot of mathematics. Over the past 14 years TEX has truly evolved into a powerful typesetting program, generally regarded as the best system in th world for handling complicated scientific text and displayed mathematical formulas, as well as producing copy near in quality and appearance to that produced by the finest traditional compositors.

Beginning around 1983, the AMS began using TEX for a small portion of its technical book and journal publishing. The AMS found TEX to be very well suited for a high volume of technical typesetting. Thus, the percentage of typesetting done in TEX increased over the next few years, until, by mid 1987, TEX was used for 100% of typeset book and journal production.

There are several factors which have contributed to TEX's huge success. Certainly one of these is the fact that unlike most proprietary composition systems, TEX can be used on almost any known computer hardware from personal microcomputers to Cray supercomputers. The fact that Knuth placed the source code for TEX into the public domain meant that any skilled systems programmer could easily tailor the TEX program to run on any computer. Also unlike many proprietary systems, the cost of most TEX implementations is reasonably low to minimal.

Another significant contribution to TEX's success is that it provided mathematicians with a tool for expressing mathematical notation in a simple text file on a computer. This allowed the mathematician to use the computer both as a means of typesetting mathematics, and for communicating mathematics to other colleagues' computers via electronic mail or other electronic transfers. It is not surprising that TEX is now pervasive throughout academia.

TEX's coding mechanism, however, sometimes appears overly complex and gives the impression to some that it would be difficult to learn. This is true if one wishes to become a devoted TEXnician and learn all of the intricacies of TEX. But TEX's powerful macro capability, which allows a single command to be an abbreviation for a complex set of commands, enables TEX programmers to develop macro packages that greatly simplify the *use* of TEX. The AMS developed such a macro package, namely $\mathcal{AMS}$-TEX, in the early 1980s. $\mathcal{AMS}$-TEX greatly simplifies the author's use of TEX to the point that simple documents can be typeset by example. In the late 1980s the AMS developed the $\mathcal{AMS}$-LATEX macro package as a result of users requesting better mathematics typesetting for the popular macro package LATEX.

The AMS became interested in TEX both for the promise of very high quality output, and for the hope that authors would one day be able to submit their files electronically to the AMS for publication. The fact that TEX is in the public domain and therefore inexpensive to obtain, as well as the fact that TEX runs on almost any computer you can name, means that TEX is widely available to a large number of people—especially mathematicians at research centers who are potential authors of technical articles or books for publication. A mathematician at a university with a PC can be writing his or her document in TEX and obtain output from a dot matrix printer or laser printer on site. When the document is finished, the TEX source file can be sent electronically to the AMS, or any other publisher. Thus, the time it traditionally takes to typemark and keyboard the document from paper manuscript is saved, and the author gains in the assurance of knowing that when proof copy is received back from the AMS, the author is reading the same file that he or she has already carefully proofread. For example, the author doesn't have to painstakingly check all of the mathematical equations for correctness because this has already been done before submission to the AMS.

3 Electronic Submissions

The goal of receiving TEX-encoded electronic manuscripts from authors has now been realized. The AMS began its electronic submission program in 1988. The number of electronic manuscripts prepared in $\mathcal{AMS}$-TEX and $\mathcal{AMS}$-LATEX has steadily increased over the past few years. In 1991 the AMS received nearly 500 electronic submissions, accounting for 16% of total submissions for that year, and this figure has risen to 22% to date for 1992 submissions. Over 75% of these electronic submissions were received via electronic mail. The rest were received on either PC or Macintosh disks sent via postal mail, with approximately twice as many PC disks as Macintosh disks being received.

The objective of the electronic submission program is to receive documents that will merge smoothly into the production stream at the AMS. For the AMS program, the key factors that have lead to realizing this goal are specific guidelines describing the preparation of an electronic manuscript, adequate author support via telephone and e-mail, and a limitation on the types of documents that will be accepted (only $\mathcal{AMS}$-TEX and $\mathcal{AMS}$-LATEX documents are accepted electronically).

It became evident that early communication with authors, preferably before writing starts, was very important. Two documents entitled *Guidelines for Preparing Electronic Manuscripts* were written in both $\mathcal{AMS}$-TEX and $\mathcal{AMS}$-LATEX versions. These documents provide the author with both general and specific instructions for the creation of the electronic manuscript, such as:

- a short description of how an electronic manuscript is processed;

- a checklist of information that must accompany a manuscript order to avoid delays;
- instructions for using the logical tags, which ones are required, and examples;
- references to more detailed manuals for the basic TEX package ($\mathcal{AMS}$-TEX or LATEX);
- instructions for submitting the file on diskette, by e-mail or by FTP;
- where to go for help.

The Guidelines do not try to show an author how to use TEX itself (or $\mathcal{AMS}$-TEX or LATEX), but only what is specific to AMS publications and built into the author's macro packages. Although electronic submissions were being received as early as 1988, no formal guidelines existed until 1990. Since the publication of these Guidelines the quality of electronic submissions has improved greatly, as measured by the number of them that require special handling and the amount of attention required from editorial and composition staff.

In addition to these guidelines, generic documentstyles for both $\mathcal{AMS}$-TEX and $\mathcal{AMS}$-LATEX are also available to the author. Both these generic documentstyles and the ones use in production at the AMS are based on generic (SGML-like) markup. In these documentstyles, the elements of a document are tagged for logical structure, not for typographic appearance. In this way an author need not know exactly what publication his or her article will appear in while writing the paper, because logical tagging provides the flexibility of moving a paper from one publication to another by simply changing the documentstyle being used.

The documentstyles provided to authors define the appearance of a "generic journal" format, whose appearance is based loosely on the *Journal of the American Mathematical Society.* Again it is important to emphasize that a paper typeset with this generic documentstyle will resemble an article from the AMS journals, but it most certainly will not be identical in appearance. The purpose of the generic documentstyle is to provide the author with a logical framework into which the document should be encoded. Both $\mathcal{AMS}$-TEX and $\mathcal{AMS}$-LATEX contain definitions for mathematical elements such as displayed and aligned equations, matrices, case statements, etc. Macros for theorems, proofs, section headings, lists, and so forth are provided by the generic documentstyle, and the author is instructed by the Guidelines that these macros **must** be used. A heading in the form `\centerline{\bf Introduction}` is unacceptable because it defines the heading's typographic appearance but not its logical structure. The proper form would be `\heading{Introduction}`.

In fact any typographic-specific TEX commands such as specific font calls, vertical or horizontal space, and page and line breaks are specifically prohibited. The reason for this is twofold: AMS publications are typeset using Times Roman fonts and the differences in character widths between Times and Computer Modern will mean different line and page breaks than what the author sees, and the editorial staff may request changes to the text itself that would affect line and page breaks.

Occasionally an author who is also a TEX expert is troubled by the restrictive posture the AMS has taken toward electronic submissions. These authors generally wish to base their submissions on a favorite macro package they have personally developed or acquired elsewhere. While a few simple definitions to act as substitutes for repeated phrases or other constructs are allowed (even encouraged), submissions which are dependent on any additional packages other than those supplied by the AMS frequently will not merge into the AMS production system without attention from a skilled TEX programmer. This does not result in the anticipated savings of time and money for such submissions.

4 AMS Production Cycle

4.1 Typeset Books and Journals

The Society's TEX-based production cycle is really quite traditional. Manuscripts are refereed and accepted for publication by an editorial board for the book or journal. Members of these editorial boards are mathematicians from academic or research sites, and are not employees of the AMS. Whether or not an author used TEX to prepare their submission has no bearing in the acceptance process.

When an accepted manuscript is forwarded by the editorial board to the Providence office, it is logged into a tracking system which will monitor the paper's progress through various correction runs, until the paper is finally typeset as part of a specific publication. Upon receipt of the paper manuscript the existence of an electronic version is determined. If an electronic version is unavailable, the paper manuscript is delivered to the Editorial Department at the AMS, where the paper is typemarked and copyedited.

Typemarking refers to the process of labeling the logical components of the paper; all headings and constructs such as theorems, definitions, proofs, etc., are clearly marked by the editorial staff. At this point the paper is forwarded to the Composition Department where it is keyed in $\mathcal{AMS}$-TEX. The composition staff consists of about 15 keyboarders, between a third and a half of whom work from home and use terminals connected to the AMS computers via modems and phone lines. The use of terminals and modems rather than PCs was deemed preferable for several reasons, such as centralization of data and ease of communication via electronic mail. Keyboarders working from home have access to the technical staff with electronic mail, and documentation and other information can easily be distributed to the at-home staff electronically.

The at-home keyboarders code the papers in $\mathcal{AMS}$-TEX and run the paper through a syntax check when

finished. The paper must successfully pass the syntax checker before it can be submitted for proof copy. These keyboarders do not have graphic terminals nor printers at their disposal, so they are never able to actually view the output of their $\mathcal{AMS}$-TEX coding. They are encouraged to key the paper as quickly as possible, and are paid by the keystroke.

Every night proof copy is generated for all papers keyed that day, and these are delivered back to the editorial staff along with the original manuscript. The proof copy is read against the orginal manuscript, and corrections are marked on the proof copy. The paper is then given back to the same keyboarder who originally keyed the paper for corrections. Payment at this stage is a small, fixed amount per page. When these corrections have been made a set of proof copy is sent to the author for approval.

In the case of an electronic manuscript, after it is logged into the tracking database the file is given to a keyboarder who changes the documentstyle being used from a generic style to the appropriate publication style. A utility is run from within the keyboarder's editor which searches the file for certain unwanted typesetting codes such as those dealing with page and line breaks, vertical and horizontal space, magnification, `\hsize` and `\vsize`, and control characters. This utility produces a report at the top of the data file listing what it located, and the keyboarder searches the file for these items and alters them as needed. The keyboarder must also check any local definitions made by the author to make sure they don't conflict with AMS definitions.

At this point proof copy is generated and sent to the editorial staff, where it is copyedited only to the extent necessary to mark for change anything that does not conform to AMS style. After these corrections are made a proof copy is sent to the author for approval.

An electronic submission can be rejected if the author has not used $\mathcal{AMS}$-TEX or $\mathcal{AMS}$-LATEX, or if they did use one of these packages but did not follow the suggestions in the Guidelines. The overriding consideration in making this decision is whether it would be quicker to alter the author's file to make it compatible with the production system, or to re-key the author's paper. A short paper (5–10 pages) submitted in plain TEX without a great amount of mathematics will invariably be quicker to re-key.

At the point in which a paper is sent to the author, the production cycle becomes identical for papers originating either as electronic submissions or as paper manuscripts. After the author has approved the proof copy and perhaps requested some alterations, corrections are made and proof is again returned to the editorial staff. At this stage the editorial staff will combine individual papers to comprise an issue of a book or journal, and from this point onward these papers will be processed as a unit.

The issue is now paged, and specific formatting instructions such as line and page breaks are inserted. No line and page breaks are allowed in any correction runs prior to this stage, since the author may request alterations which would affect paging. These specific formatting instructions are held to a minimum, however, and the camera copy is sometimes altered by hand in preference to additional TEX runs.

4.2 Author-Prepared Books

The AMS also chooses to publish certain books directly from camera copy supplied by the author as a cost saving device, and to speed publication. The popularity of TEX among academic and other research centers has allowed for the production of these low-cost books with little sacrifice in quality.

Aside from the generic documentstyles described above, the Society also distributes $\mathcal{AMS}$-TEX and $\mathcal{AMS}$-LATEX documentstyles for a few specific book series which publish author-prepared copy. Authors are encouraged to use these documentstyles in the preparation of their work, and submission of both the TEX source file and the DVI file is requested. The DVI file is simply spooled to a typesetter, and the TEX source file is archived for possible future use. If the author has followed the suggestions in the Guidelines and used the package correctly, the final result of books published in this fashion will have the appearance of a comparable book produced via the traditional route, with the exception that the author-prepared work will be published in Computer Modern fonts while all other AMS publications use Times Roman.

5 Future Directions

5.1 Production System Enhancements

Several major enhancements to our current production are either under way or being planned for the next few years. The TEX macros which comprise the current system have been altered many times by many different people over the past decade. A study last year revealed that somewhere around 50% of the code in the current system was either obsolete or never executed. A rewrite of the system is currently underway.

One major difficulty that the rewrite will address is the ability to easily produce a set of typographic specifications for each publication. The Editorial Department has a set of written specifications for each publication, but it is a difficult task to ensure that the TEX macros which produce the publication exactly match the written specifications. Work is being done to have the TEX macros actually read a file containing the typographic specifications for each publication.

A prototype of this file is being developed. Every logical component of a book or journal article is listed in this file, along with such attributes as the font to use, the amount of space (maximum, minimum, and optimal) before and after the element, penalties for page

breaks before and after, other attributes such as linespacing, wordspacing, case, justification, etc., are listed in the file in a format that is both human readable and can be parsed by TEX.

Another problem area being worked on is the ability to change a documentstyle for a given publication without affecting papers for that publication which are already in the production stream. Changes to publications are sometimes requested to begin with a specific issues (such as the first issue of a new year), but when papers are accepted for many journals, they are not accepted for a specific issue but instead go into a pool for that journal. Therefore, it is often the case that papers for several different issues of a journal are already in process at various points along the production stream at any given time.

The idea here is to embed a date into the data file which TEX will read. In addition, whenever changes are made to any macro files in the production system, a copy of the code is first copied to the bottom of the macro file before any changes are made. The unaltered code is marked with the date the revisions are being made to the file, and then changes to this code are made in the main section of the file. When a data file is processed, TEX reads the date in the data file and then reads the macro file(s). After the main section of the macro files are read, TEX checks the dates on the altered code at the bottom of the macro file, and stops reading when the date in the macro file is less than the date in the data file.

For example, supposed `\heading` is to be changed from boldface to italic type, and suppose there are two data files, one with a date of 1-MAY-1992 and the other with 1-JUN-1992. Suppose also that the change to `\heading` was made on 15-MAY-1992. The desired result is that the data file keyed on 1-MAY-1992 should retain boldface headings, while the 1-JUN-1992 should pick up the new italic headings. The diagram below shows how the above scheme accomplishes this.

macro file: `\def\heading#1{\noindent\bf#1}`
(additional code)

`\dateofchange{15-MAY-1992}`
`\def\heading#1{\noindent\it#1}`

data file #1: `\filedate{1-MAY-1992}`
ITALIC Headings
data file #2: `\filedate{1-JUN-1992}`
BOLD Headings

5.2 Changing Fonts: CM, Autologic Times Roman, PostScript Times Roman

The STI system used during the late 1970s and into the early 1980s typeset the AMS publications using Times Roman fonts. For a period of several years during the mid 1980s some AMS journals were published which contained articles typeset with both STI and TEX in the same issue. The enthusiastic move toward a TEX-based production system was slightly diminished by the lack of TEX-compatible Times Roman fonts at that time. Many people involved in the publications process felt that Times Roman had become an industry standard for technical publications for its readability and appearance, and they were somewhat reluctant to publish using TEX's Computer Modern fonts.

In 1987 the first Autologic APS-μ5 was purchased, and work was immediately begun to create a set of TEX-compatible Autologic Times Roman fonts. Accessing the Autologic Times Roman, Italic, and Bold fonts were fairly straightforward, but the creation of an Autologic Times Math Italic font for use with TEX was a major effort. The details of this work were presented at tenth annual TEX User's Group meeting at Stanford University in August of 1989. A published version of this presentation, entitled *Migration from Computer Modern Fonts to Times Fonts* by R. E. Youngen, W. B. Woolf, and D. C. Latterner can be found in TUGboat, volume 10, number 4.

At the time when the APS was purchased, high-resolution PostScript typesetters were available in the marketplace, but they were all much too slow for the volume of production at the AMS. However, since that time the hardware has improved dramatically, and PostScript has assumed its place as an industry standard. The Society is now preparing to move book and journal production to PostScript fonts over the next few years.

Switching production to PostScript Times fonts is contingent on a PostScript Times Math Italic font, equivalent to the Autologic version in current use. The *MathTime*$^{\mathrm{TM}}$ fonts recently released by Michael Spivak contain such a font, and plans call for this to be evaluated along with the possibility of the AMS creating its own such font.

The move to PostScript is driven not only by its status as an industry standard, but also for the promise of such things as auto-inclusion of PostScript graphics into TEX documents and the ability to output completed pages directly to film negatives or plate materials. Authors' graphics are currently re-rendered on a Macintosh using Adobe Illustrator, which is a PostScript-based application. These graphics are currently being output to the Society's PostScript imagesetter and stripped onto the camera copy of the TEX document by hand.

By typesetting completed pages, substantial cost savings may be realized if typesetting directly on plate materials proves practical. The current cycle is to produce camera copy from a typesetter which is taken to the print shop and photographed to produce a negative. The negative is then used to make a metal plate that can be placed on the press. Typesetting directly to plate material would save in both labor and materials costs for the intermediate steps.

5.3 Electronic Dissemination/Publication

Dissemination of information by electronic means is not a new concept to the AMS. Since 1982, *Mathematical Reviews (MR)* has been distributed electronically as well as on paper, with the data being installed in the public databases of vendors such as Dialog and ESA. Prior to 1985, MR was typeset using the STI system, and the data had to be converted into a human-readable form for database installation. Effective with the January 1985 issue, TEX became the sole input language. Since then, a set of macros has been developed to permit database users to download items from the databases to their local computer, process it with TEX, and print out the result, which is for all practical purposes identical to the published review. In 1989 MR was also released on CD-ROM which is accessible using search software on PC-compatible and Macintosh computers. Full reviews from 1980 to the present are now available on CD-ROM, along with bibliographic information from 1940–1979.

Beginning this year, the *Bulletin of the American Mathematical Society* is also being distributed electronically via the Society's e-MATH node (e-MATH.ams.com) on the Internet. The *Bulletin* is a free journal as a privilege of membership and $\mathcal{AMS}$-TEX files for this journal are installed on e-MATH for public access. Files for this journal are prepared in the same way as for any other journal, and a paper copy is still produced. Experience with this service will help determine the direction of other similar experiments.

5.4 Standard Generalized Markup Language (SGML)

The AMS has already concluded, independent of the SGML initiative, that generic markup is the only sensible way to approach electronic storage and processing of manuscripts and bibliographic data. Though strict SGML syntax is not used in the production of AMS publications, the underlying principles of SGML are in practice.

The Ann Arbor office is developing an SGML-based software tool which will perform several important document-handling functions, such as:

- accept an $\mathcal{AMS}$-LATEX file from an author, parse it into an SGML document, and store it in a document database which recognizes and manages the structural parts of the document;
- accept an SGML file and produce an $\mathcal{AMS}$-LATEX file which can be typeset through TEX;
- allow several people (cooperating authors, editors, etc.) to work together (from remote sites) on the creation of a document, maintaining *version control*, which will allow authors, editors, etc., to track the modifications each has made to the file;
- allow referees, editors, reviewers, and readers to make comments about the file, which will be stored as *annotations* attached to specific points or regions in the document, with identification of the commentator and a record of the time of annotation.

The expectation is that this tool will someday allow for on-line editing in the production of the AMS publications, replacing the cumbersome process of creating paper galleys for each correction run. Instead of corrections being marked on paper with colored pencils by an editor, then keyboarded by a typist, and finally re-proofread by an editor, the editor will be able to make corrections directly in the file and see the results instantly on the screen.

Perhaps even more importantly, this tool will define an environment for true collaborative development of electronic and paper publications by authors situated at remote sites throughout the world. *Version control* and *annotation* are the key capabilities required for a tool appropriate to collaborative research and writing in mathematics.

6 Conclusion

It is impossible to predict to what extent electronic publishing will replace traditional publishing on paper. While the advantages of electronically stored and produced documents are many and conclusive, the affinity for paper publications may be difficult to overcome. As computing devices become smaller and more portable and are linked by radio networks, the difference between sitting beside the fireplace with a book you pulled off the shelf and a book reader which electronically retrieved your favorite text becomes less obvious. TEX and SGML, or some offshoots thereof, are well positioned to play an important role in this future.

Standard dtd's and Scientific Publishing*

N.A.F.M. Poppelier†

and

E. van Herwijnen‡

and

C.A. Rowley§

August 1992

Abstract

This paper has two parts. In the first part we argue that scientific publishing needs *one* standard dtd for each class of documents that is published, for example one for all research papers and one for all books. In the second part we apply this reasoning to mathematical formulas, and we outline some design requirements for a document type definition for mathematical formulas. In the appendices we discuss and compare existing document type definitions for mathematical formulas.

1 Introduction

In the preface to [1] Charles Goldfarb wrote that the Standard Generalized Markup Language can be described as many things, and that SGML is all that – and more. In the introduction to [1] Yuri Rubinsky wrote:

> *ISO 8879 never describes SGML as a meta-language, but everything about its system of declarations and notations implies that a developer has the tools to build exactly what is required to indicate the internal structure of any type of information in a common tool-independent manner.*

Indeed, a strong point of SGML is that it can be regarded as a meta-language, a tool with which one can define the syntax of many languages, very much similar to context-free grammars. In SGML terminology these 'languages' are called document type definitions, called *dtd's* for short. Dtd's can be written for any type of information, e.g. research papers, books and music. A dtd can be used for many purposes, of which two important ones are storage and exchange of information coded according to this dtd.

The premise of this paper is that the exchange of information, if it is based on SGML, needs a single common dtd, agreed upon by all parties involved, for each class of documents that is exchanged.

Suppose two parties, A and B, exchange information in the form of one class of documents, and that they each have a dtd, D(A) and D(B), with D(A) not identical to D(B). If A sends a document to B then A can include the document type definition, D(A), for that document (instance) at the beginning of the document. This enables B to use an SGML parser to check the validity of the document he received. However, there is nothing more B can do with the document: the dtd D(A) contains no information about the *meaning* of the coding scheme that D(A) defines, and a mapping of the document from D(A) to D(B) is a procedure that cannot be automated. The problem becomes even more difficult when a third party, C, is introduced, who accepts material from both A and B. How is C going to handle material with two different coding schemes?

This is where we encounter one of the weaknesses of SGML *as it is being used currently*, namely that it enables every party involved in this process to define and use a different dtd.

2 Scientific publishing

In the rest of this paper we concentrate on the exchange of information that occurs in scientific publishing, in particular on the exchange of papers that contain math-

*Presented at the 9^e NTG meeting, June 4, 1992, Amsterdam. To be published in EPSIG news.

†Elsevier Science Publishers, P.O. Box 2400, 1000 CK Amsterdam, the Netherlands.

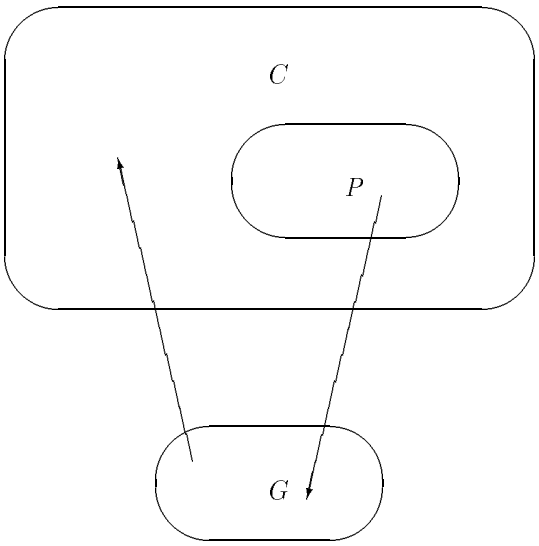
‡CERN, 1211-CH Geneva 23, Switzerland.

§Open University, 527 Finchley Road, London NW3 7BG, UK.

ematical formulas and are published in research journals. Recent developments in this area formed the main reason for writing this paper.

A few standards for encoding of mathematical formulas have already emerged, of which a well-known one is the AAP Standard or Electronic Manuscript Standard [2]. A dtd for mathematical formulas accompanies this standard, but it is not part of it. Another standard for mathematical formulas is the one adopted by CALS [3], and others are under development [4, 5].

The handling of mathematical formulas in scientific publishing is part of the bigger whole of information exchange within a (the) scientific community, with the publisher as intermediary, as is shown in figure 1.



Figuur 1: *Information exchange within a (the) scientific community.*

The authors of research papers are the providers, P. The publishers are the gatherers of information, G. They accept information from many providers, gather this in the form of a journal issue, and distribute this. In this process, the publisher provides a quality check via the system of peer reviewing, makes notation consistent, and in some cases improves the prose. The information is distributed to a group of consumers, C, with the set C a superset of the set P. In this process, two sorts of information can be exchanged:

- material that is structured in the sense of being encoded according to, and checked against, some formal structural specification such as a dtd;
- material that is not structured.

At present most of the material exchanged in the process of scientific publishing is of the unstructured type. We expect that this will remain the situation in the near future. As soon as authors get the possibility of using more sophisticated tools, we expect that publishers will receive increasing numbers of papers of the structured type.

Several scientific publishers, among whom Elsevier Science Publishers, have adopted SGML as the future main tool for the process of publishing scientific articles [6], and several other publishers have made, or are expected to make, the same choice. The European Laboratory for Particle Physics (CERN), a large community of information providers, are using SGML to automate the loading of bibliographic information in their library's database [7]. For both authors and publishers it would be advantageous to agree on one dtd for the encoding of research papers. There are several reasons for this:

- Most authors do not submit all their articles to one and the same publisher every time. At present they are confronted with 'Instructions to Authors' that differ significantly from publisher to publisher.
- A recent trend is that authors prepare their papers with text-processing software on some computer. This enables them to send the paper in electronic form (electronic manuscript or 'compuscript') to the publisher. Publishers are confronted with a variety of text-processing software on a variety of computer systems [8, 9]. Moreover, every field of science appears to have its own 'Top Ten' of most used text processing packages.
- Bibliographic information about all research papers in all (or most) scientific journals is stored in bibliographic databases.

In an ideal world, authors would still be able to use their favourite text-processing system, which would generate SGML 'behind the screens', so to speak. All publishers would accept one standard dtd, and all text-processing systems would be able to generate documents prepared according to this dtd, and all bibliographic databases would be able to store this material.

An example of activities towards achieving this ideal situation: the European Working Group on SGML (EWS) and the European Physical Society (EPS) have taken the Electronic Manuscript Standard and are trying to develop it into a complete dtd, which should be acceptable to information providers, information gatherers and information consumers. The Electronic Manuscript Standard is now a Draft International Standard, ISO/DIS 12083. The EWS and EPS hope that the final standard will include their work.

3 Encoding of mathematical formulas

In Annex A of ISO 8879 [10] we find the following:

> *Generalized markup is based on two novel postulates:*
> 1. *Markup should describe a document's structure and other attributes rather than specify processing to be performed on it, as descriptive markup need be done only once and will suffice for all future processing.*
> 2. *Markup should be rigorous so that the techniques available for processing rigorously-defined objects like programs and databases can be used for processing documents as well.*

There is no reason why this should not be valid for mathematical formulas. We need to delimit the kind of mathematical formulas we are trying to describe if we want an unambiguous structure. The field of mathematics is so vast, that it may be impossible to design a single dtd that covers every kind of mathematical formula. If we concentrate on those sciences which use mathematics as a tool, for example physics, we see that the mathematics used in many physics papers can be described as "advanced calculus". This definition can be made more precise by referring to some standard textbooks containing these types of formulas, e.g. *Handbook of Mathematical Functions* [11] and the *Table of integrals, series and products* [12].

If we aim for rigorous encoding of mathematical formulas (the second postulate), we must develop a system of descriptive markup of mathematical formulas that enables us to:

- convert the formulas between different word processors;
- store the formulas in and extract them from a database;
- allow programs to input or output formulas in descriptive markup.

An example of the first application would be the conversion of mathematical formulas coded in LaTeX to, say, Word[1] via SGML. The benefits of using SGML as an intermediate language for conversion are described in [13]. Note, for example, that the number of programs required for pairwise conversion between n languages is proportional to $n^2 - n$ without an intermediate language, but to $2n$ with an intermediate language.

An example of the second application would be encoding and storing the complete contents of the above mentioned *Handbook of Mathematical Functions* [11] and *Table of integrals, series and products* [12] in a database, so that this information can be accessed on-line by, say, mathematicians and physicists. Many articles have mathematical formulas in their titles, so any program that extracts bibliographic data should be able to handle mathematics as well.

An example of the third application would be the extraction and subsequent use in a computer program, written in an ordinary programming language or, for example, in Mathematica.[2]

At this point we come back to the ideal world for scientific publishing we sketched earlier. In this world, publishers would use one standard dtd for scientific papers, which enables them to prepare a primary publication – in paper and (or) in some electronic form – and to store the information in databases for various secondary purposes.

The question now is: what should a dtd for mathematical formulas look like, if it is going to be used for these purposes?

There are two choices for a dtd for mathematics:

- P-type: the dtd reflects the Presentation or visual structure, Examples of this type are discussed in the appendices.
- S-type: the dtd reflects the Semantics or logical structure At present no dtd's of this type exist.

The quotation from Annex A of ISO 8879 [10] indicates the preference of the creator(s) of SGML: markup of a formula should be of S-type, it should describe the logical structure of the formula, rather than the way it is represented on a certain medium, say the page of a traditional (non-electronic) book.

Let us suppose, for the sake of the argument, that an information gatherer, a publisher, chooses a dtd of S-type. This raises two further questions:

1. Is descriptive markup of mathematical material possible?
2. If it is possible, who can use it and for which purposes?

The second question needs some explanation. As discussed in section 2, in the process of scientific publishing two sorts of information can be exchanged: mathematical material that is structured according to

[1] Word is a registered trademark of MicroSoft.

[2] Mathematica is a registered trademark of Wolfram Research, Inc.

a formal structural specification, and material that is not structured. This means that there are two possible scenarios.

Scenario 1: an author submits a paper in the form of a manuscript (paper), i.e. with unstructured formulas, or a compuscript with mathematical formulas in P-type notation (TEX, WordPerfect, ...).

Scenario 2: an author submits a paper with mathematical formulas in S-type notation.

In scenario 1 it is the task of the publisher to convert from paper or P-type notation to S-type notation. Before we discuss the feasibility of this conversion, we will first look at some characteristics of mathematical notation.

3.1 Characteristics of mathematical notation

Mathematical notation is designed to create the correct ideas in the mind of the reader. It is *deliberately* ambiguous and incomplete: indeed, it is almost meaningless to all other readers. Or, more technically: the intrinsic information content of any mathematical formula is very low. A formula gets its meaning, i.e. its information content, only when used to communicate between two minds which share a large collection of concepts and assumptions, together with an agreed language for communicating the associated ideas.

The ambiguity encountered in mathematical notation can be of two types [14]:

1. A generic notation uses the same symbols to represent similar but different functions, for example '$+$' or '$\times$'. In the case of addition this is not really a problem, but multiplication *is* a problem since, multiplication of numbers is commutative, whereas matrix multiplication is non-commutative!
2. A more fundamental ambiguity is posed by the same notation being used in different fields in different ways. For example: f' stands for the first derivative of f in calculus, but can mean 'any other entity different from f' in other areas.

More examples of ambiguity are:

- Does $\bar{x}$ represent a mean, a conjugation or a negation?
- Is i an integer variable, e.g. the index of a matrix, or is it $\sqrt{-1}$?
- The other way around: is $\sqrt{-1}$ denoted by i or by j?[3]
- What is the function of the 2 in SU_2, $\log_2 x$, x_2, x^2, T_2^2?[4]
- Is $|X|$ the absolute value of a real (complex) number X or the polyhedron of a simplicial complex X [15]?

The inverse problem, which is equally common, arises when different typographical constructs have the same mathematical meaning. For example, the meanings of both the following two lines would be coded identically

$$3 + 4 \pmod 5$$
$$3 +_5 4$$

and this would lead to great difficulty if an author wanted to write:

> We shall often write, for example, $3 + 4 \pmod 5$ in the shorter form $3 +_5 4$, or even as simply $3 + 4$ when this will not lead to confusion.

Of course, natural languages are similarly ambiguous and incomplete, but no one we know is suggesting that in an SGML document each word should be coded such that it reflects the full dictionary definition of the meaning which that particular use of the word is intended to have!

3.2 Who performs the markup of math?

How does one convert P-type mathematical material, which an author has produced, to S-type notation, which the publisher uses?

In [1, p. 9] Goldfarb gives a three-step model for document processing:

1. recognition of part of a document (adding a generic identifier for the appropriate element);
2. mapping (associating a processing function with each element);
3. processing (e.g. translating elements into word processor commands).

In the publishing of scientific papers and books steps 2 and 3 are the responsibility of the publisher. Traditionally, step 1 was also their responsibility: the technical editor adds markup signs in the margin of the manuscript, depending on the text and the visual representation that the house style dictates. It is, however, unlikely that a technical editor is capable of identifying the precise function of every part of a mathematical formula, for several reasons, most of which were discussed in the previous subsection, namely that mathematical notation:

- is not unambiguous,
- is not completely standardized,
- is not a closed system.

Even if the technical editor were capable of identifying every part of a formula, this would be too time-consuming – and therefore too costly. However, under certain conditions [16], automatic translation from

[3] There are examples of authors actually writing something like $[L_i, L_j] = \frac{i}{2} L_k$, where the first i is an index, and the second i stands for $\sqrt{-1}$.

[4] In SU_2 it is the number of dimensions of the Lie group; in $\log_2 x$ it is the base of the logarithm; if x is a vector, the $_2$ in x_2 is an index; the ^{2} in x^2 could be a power, but if T is a tensor, the ^{2} in T_2^2 is a contravariant tensor index.

visual structure to logical structure of mathematical material is simplified greatly.

This, and what we discussed in section 3.1, leads us to conclude the following. A publisher has no choice but to use a P-type dtd for mathematical material that is submitted in unstructured form or in P-type notation. Even if S-type markup of a mathematical formula would be possible, conversion from P-type to S-type would be difficult or even impossible. Conclusion: the tags for S-type markup should not be added by the information gatherer, but by the information providers, i.e. the authors, who should be able to identify each part of their formulas.

3.3 Feasibility of S-type notation

In our second scenario, authors would submit papers with mathematical formulas in S-type notation. This would enable the publisher to 'down translate'[5] to any mathematics typesetting language (P-type notation). However, the same reasoning as in section 3.1 leads us to the following conjecture:

Conjecture. It is impossible to create an S-type dtd for *all* of mathematics.

Representing the "full meaning" of a mathematical formula, if such a notion exists, will almost certainly lead to attempts to pack more and more unnecessary information into the representation until it becomes useless for any purpose. This is rather like Russell and Whitehead reducing "simple arithmetic" to logic and taking several pages of symbols to represent the "true meaning of $2 + 2 = 4$".

Even if it were possible to define an S-type dtd for a certain branch of mathematics, this still gives problems. Supposing an S-type dtd contains an element for a "derivative" of a function. Since the S-type dtd will not contain any presentational attributes, a decision will have to be made to represent the derivative of $f(x)$ on paper as $f'(x)$ or $\frac{\mathrm{d}f(x)}{\mathrm{d}x}$. There are, however, times (such as in this article) that both representations are required for the same semantic object, and that the author will need other notation in addition to that defined by the S-type dtd.

A likely reason for the belief that an S-type dtd is possible, is that many people in the worlds of document processing or computer science are convinced that each symbol has at most a few possible uses and that mathematical notation is as straightforward to analyse as, for example, a piece of code for a somewhat complicated programming language. The reality is that mathematical notation is more akin to natural language: it is ambiguous and incomplete, as we pointed out earlier.

3.4 Some problems with existing languages

To show that it is not obvious to capture mathematical syntax in a dtd, let alone its semantics, consider the example of a limit

$$\lim_{x \to a} f(x).$$

The syntactical structure of a limit is:

- The limit operator
- The part containing the variable and its limit value
- The expression of which the limit is to be taken

The first part could:

- always be "lim", in which case it is just a part of the presentation of the formula and it should be left out.
- be one of a finite list of alternatives, indicating the type of limit (lim inf, sup, max etc.). In this case it should be an attribute.
- be any expression.
- be any text.

We think the second possibility comes closest to the syntax of the limit construct. The second and third parts can be any mathematical expression.

Now let's look at the way this formula is coded with the dtd's from ISO TR 9573, AAP math and Euromath respectively. Using the mathematics dtd from ISO TR 9573 there are three possibilities:

- `lim <sub pos=mid> x &rarr; a </sub> f(x)`
- `<plex><operator>lim</operator><from>x &darr; a</from> <of>f(x)</of></plex>`
- `<mfn name=lim><sub pos=mid>x &rarr; a</sub><of>f(x)</of></mfn>`

The AAP math dtd strongly suggests the following representation:

```
<lim><op><rf>lim</rf></op><ll>x \&rarr;
            a</ll><opd>f(x)</opd></lim>
```

whereas with the Euromath dtd we would have:

```
<lim.cst><l.part.c limitop=limm><range>
                  <relation>x \&rarr; a
</relation></range></l.part.c><r.part.c>
                 <textual>f(x)</textual>
```

We see that the AAP and Euromath expressions are closest to the limit syntax. The best solution from ISO TR 9573 involves a more general "plex" construct, which can be used for integrals, sums, products, set unions, limits and others. When the plex construct contains the actual lower and upper bounds it may even give semantic information.

Some mathematicians, however, are not satisfied with this solution [17]. The plex operation is probably a notation for an iterated application of a binary operation (e.g. sums and products), while limits are of a

[5] 'Down' because information is lost in the process; we borrowed the terminology of translating 'up' and 'down' from Exoterica OmniMark.

different nature. In many cases only the from part will be used, and there the whole range of the bound variable will be indicated, as an interval or a more general set. How does one go about extracting the bound variable?

This supports our conjecture from the previous section, namely that it is very hard to capture the semantics for all mathematics. It also suggests that some redundancy is required to select whichever notation is most appropriate in a certain context.

4 Re-using mathematical formulas

There are two important uses for a generically coded mathematical formula. The first one is in a mathematical manipulation – or computer algebra – system (MMS), such as Mathematica [18] or Maple [19]. Computer programs for the numerical evaluation of formulas, for example written in FORTRAN or Modula-2, can also be regarded as mathematical manipulation programs.

The second form of re-usage is in a mathematical typesetting system, for formatting the formula on paper or on screen; examples of this are TEX [20] and eqn/troff [21, 22].

For computer algebra systems the notation for the formula should be such that a particular type of manipulation on a particular system is possible, given a 'background' of concepts and assumptions that enables the system to interpret the input as a mathematical statement.

The coding of a formula that is adequate for document formatting, for example the TEX notation `f^{(2)} (x)`, is very unlikely to contain much of the information required for a manipulation system to make use of it. However, for a limited field of discourse it is feasible to use the same coding for both types of system [16].

Some examples: the square of $\sin x$ is typographically represented as $\sin^2 x$, but a system like Mathematica or Maple would probably prefer something like $(\sin x)^2$ as input. Typesetting the inverse of $\sin x$ as $\sin^{-1} x$, however, could be confusing: does it mean $1/(\sin x)$ or $\arcsin x$?

An MMS would probably require the second derivative of a function f with respect to its argument x to be coded as $(D, x)((D, x)\, f(x)))$, but on paper this would be represented as $f''(x)$, or $f^{(2)}(x)$, or

$$\frac{\mathrm{d}^2 f(x)}{\mathrm{d}x^2}.$$

On the output side of a MMS there are other problems since some of the coding necessary for typographically acceptable output cannot be automatically derived by the system from the coding used by the MMS.

The Euromath view [17] is that a common interface should be designed together with the manufacturer of a MMS. Perhaps an MMS-type dtd will be required.

5 Related problems

Another problem is, of course, that mathematics is by its nature extensible, so there will always be new types of manipulations to be done. Notations are changed or new notations are invented almost every day, figuratively speaking. Normally these new subjects will use existing typographic representations, but the computer algebra system will not know what formatting to use! Occasionally a new typographic convention will be needed. And although there is agreement on the notation for most mathematical concepts, authors of books on mathematics tend to introduce alternative notations, for instance when they feel this is necessary for didactic reasons. Mathematical notation is not standardized, and it is open—anyone can use it, and add to it, in any way they wish.

If we consider a given dtd at any time, we have to ask ourselves: can an author add elements when the need for this arises? Theoretically the answer is 'Yes, he can' [23, p. 71], although it is not straightforward to include the new elements in the content models of existing elements.

Are such modifications by the author desirable? A dtd which is locally modified by an author will quickly give rise to the situation described in the introduction to this paper, and this should therefore probably be discouraged. Others, however, have also noticed a need for private elements, as described in EPSIG News 3, #4: one of the challenging aspects of using SGML being encountered by the TEI is that the guidelines need to be extensible by researchers. They need to be able to extend the dtd's in some disciplined way [24].

This problem, however, may not be a serious one. The collection of style elements is almost a closed set, since the number of fonts, symbols and ways to combine them is limited. In fact, most notation is not syntactically new, since the limited number of constructs works well as a notation. The multitude of notations is obtained by combinations of fonts, symbols and positions (left or right subscript, left or right superscript, atop, below, ...), and by giving one notation more than one meaning. This again seems to support our view that only a P-type dtd can be constructed for *all* of mathematics.

An SGML dtd, of whatever type, also doesn't solve the problems of new atomic or composite symbols, which occur frequently in mathematics. As with new elements, an author can add entities for these new symbols. There is no method to add the name of a new symbol, whether atomic or composite, to an existing set of entity definitions for symbols, other than to contact the owner of the set and wait for an update.

Although there is now a standard method to describe that symbol's glyph (shape) [25], it is not practical for

an author to include it. A compromise solution seems to be to extend an existing set, such as the one from ISO [26], as much as possible, and try to standardize its use.

6 Conclusions

We have argued as follows:

- That a logical dtd in the sense of describing the structure of the mathematical meaning is as impossible for maths as it is for natural language, and also it is useless for formatting since the same mathematical structure can be visually represented in many different ways. The correct one for any given occurrence of that structure cannot be determined automatically, but must be specified by the author.
- That what needs to be encoded for formatting purposes, is information that enables a particular set of detailed rules for maths typesetting to be applied. This could be described as a 'generic-visual encoding' or 'encoding the logic of the visual structure'. To establish exactly what these codes should be will require an expert analysis (probably involving expertise from mathematicians, particularly editors, and from typographers aware of the traditions of mathematical typesetting).
- That this is different to what needs to be encoded for use in mathematical manipulation software. Since neither of these encodings can be deduced automatically from the other, a useful database will need to store both. Perhaps a separate dtd will be required to enable this communication.

Possible solutions are

- A dtd based on a hybrid of visual structure and logical structure
- Two dtd's, one for visual structure and one for logical structure, that are linked in some fashion
- Two concurrent dtd's, one for visual structure and one for logical structure.

The simplest solution is probably to have a basic visual structure which is described as an SGML entity, supplemented with a (redundant) logical structure, described by a second SGML entity. This solution avoids any special SGML features and gives the user all flexibility for mixing and matching as required.

We believe that similar reasoning can be applied to tables and chemical formulas, where the problem of separation form from content is just as complex, or even more.

A Existing mathematical notations

A.1 Comparison of existing dtd's

In making comparisons between existing dtd's we shall refer often to what is probably the best-known system for coding mathematical notation in documents. This is the version of TEX coding used in LATEX [27] (which differs little from Knuth's Plain TEX notation described in [20]), now a de facto standard in many areas. It is a mixture of visual and logical tagging, with a bias towards the visual which probably results from reasoning similar to that in this paper.

The following document type definitions for mathematical formulas were investigated for this paper: AAP [28], ISO [29] and Euromath [5].

We will try to give a few general characteristics of each of them:

AAP This dtd shows a hybrid of visual and logical tagging. It is quite similar to the mathematical notation of TEX [20].

Integrals, sums and similar constructions have subelements tagged explicitly as lower limit, upper limit and integrand (summand, ...).

The same goes for fractions, roots, and limit-like constructions.

All rectangular schemes of mathematical expressions, e.g. matrices and determinants, are tagged as 'array' in this dtd. The delimiters are not part of the construction, although matrices are usually indicated by $(\cdot)$ or as $[\cdot]$, and determinants as $|\cdot|$ Alignment of rows, columns and cells is indicated by attributes, even though they have nothing to do with function, but are in fact processing information. This idea also appears in the array notation of LATEX [27].

A subscripts or superscript is indicated as such, and not as power, index, ... Greek letters, italics, emphasized letters are all specifically marked up, which could cause ambiguity as regards the semantics of a given symbol.

It contains tags of a non-presentational nature, for example for vectors, dyads and tensors. It is possible, however, to use font or style commands to obtain the same result. Therefore, these tags, and similar ones, appear to be superfluous.

For advanced calculus, this dtd is as complete as is required by many papers. To capture the semantics of calculus, some modifications are required.

Euromath This dtd evolved from a earlier dtd in the Grif [30] system and provides a hybrid of visual and logical markup. One of the design principles was the possibility of conversion to and from mathematical notation in LATEX [27]. This could be dangerous, since the design of a dtd is a modelling activity, and the data being modelled is mathematics, not a text formatter. An immediate consequence is that the semantics that can be associated to a formula is that which is present in LATEX (which excludes tensors). It does, however, show the importance of LATEX for scientific publishing. Another consequence is that the Euromath dtd bears a strong similarity to the AAP dtd.

Alignment of rows, columns and cells is not indicated at all. However, every expression or sub-expression carries an attribute that expresses the 'style' of the expression, a concept borrowed from TEX [20]. This style specifies, among others, the placement of limits of integrals and sums, and the position and size of superscripts and subscripts. The reason for adding this attribute is unclear, since it can be derived from the context.[6]

In the Euromath dtd, a formula is considered as a sequence of parts or *constructions* that are arranged horizontally one after the other, and aligned along their reference axes. This reflects the fact that most formulas can be read aloud, e.g. over the phone, and in that sense are almost one-dimensional. Modes of mathematical expression that are clearly two-dimensional, such as commutative diagrams – see [31, p. 125] or [15, p. 312], are not covered by this dtd. These are tacitly assumed to be part of a higher level dtd.

ISO This dtd is designed for logical tagging, covering mathematical formulas from advanced calculus.

There is redundancy in its notation. For example, elements exist for tensors and their indices, but also for superscripts and subscripts. A good WYSIWYG formula editor should help authors to unambiguously markup their mathematics and to preserve its semantics.

There are tags of a non-presentational nature, for example for vectors and tensors. However, the vector tag can only be used for the object as a whole, and it is not clear how an individual component should be coded. Furthermore, this dtd contains tags for superscripts and subscripts, thus allowing – but not forcing! – a user of this dtd to write 'the square of x' as `x<sup>2</sup>` instead of `<power>2<of>x</power>`.

The ISO dtd has a large overlap with the AAP dtd, as is shown in appendix B. Both dtds can be used as an intermediate language for conversion between word processors. To capture the semantics of a an arbitrary calculus formula, more is required.

B Comparison between ISO TR 9573 and AAP math dtd's

B.1 Formula and formula reference

Note: ISO assume spaces are ignored by the text formatter and that positioning is done by according to the rules of mathematical typesetting. AAP have a NOTATION attribute on their formula that would allow blanks etc. to be ignored, and various characters recognized as operators.

ISO	AAP equivalent	Difference
Inline formula		
`f`	`f`	geo.form NOTATION (AAP)
Display formula		
`df`	`fd`	geo.form NOTATION (AAP)
`df id=`	`la`	
`df align=(left|right|center)`	`la pre post`	1. `align=center` (ISO)
		2. `la` element (AAP) allows multiple numbers
`df num=`	`contents of la element`	
Display formula group		
`dfg`		AAP has no
`dfg id=`		display formula groups
`dfg align=(left|right|center)`		
`dfg num=`		
Formula reference		
`dfref refid=`	`lar`	`lar` element has content; `dfref` is empty
`dfref page=`		the page attribute (ISO) adds the page number

[6] TEX does this too, but the user of TEX can override the program's automatic choice.

B.2 Formula content

ISO	AAP equivalent	Difference
Horizontal and vertical alignment		
`mark id=`	`hmk id=`	identical
`markref refid=`	`hmkr rid=`	
	`vmk id=`	vertical alignment
	`vmkr rid=`	is absent in ISO
Division points in a formula		
`break type=`	`tu`	Both elements are empty. `type=` indicates optionality. `tu` is excluded from various constructs.
Superscripts and subscripts		
`sub`	`inf`	
`sub pos=(pre|mid|post)`	`inf loc=(pre|post)`	mid value (ISO) missing in AAP
`sup`	`sup`	
`sup pos=(pre|mid|post)`	`sup pos=(pre|post)`	mid value (ISO) missing in AAP
Boxes		
`box`	`box style=`	style atribute (AAP) to to change rule type missing in ISO
Over embellishments		
`ov`	`a`	
`ov pos=`	`a valign=`	
`ov type=`	`ac`	`ac` element (AAP) allows simultaneous embellishments
`ov style=`	`ac`	`style` attribute (ISO) can be achieved with `ac`
Tensors		
`tensor`		the tensor element(ISO)
`tensor suffix=`		is absent in AAP.
`tensor posf=`		can be visually achieved with `sup`, `inf` and `zw` (zero width character, absent in ISO)
Functions		
`mfn name=`	`rf`	name attribute (ISO) has a finite list of values.
`fname`		`fname` (ISO) indicates
`of`		arbitrary roman function; of the argument; `rf` (AAP) only marks up the function name
Roman and italic fonts		
`roman`	`rm`	equivalent
`italic`	`it`	
Vectors		
`vec`	`v`	equivalent
Fractions		
`frac`	`fr`	equivalent

`numer`	`nu`	
`over`	`de`	
`frac` `align=(left|right|center)`	`fr align=(r|l|c)`	equivalent
		AAP has shape attribute, `fr shape=(case|built|sol)`, to indicate shape of fraction; missing in ISO
Derivatives		
`diff`	`inc`	The `inc` (AAP) element
`diffof`		is for increments. The type
`by`		is given in the content,
`diff type=partial`		but it is not marked up
General plex (limits)		
`plex`	`lim`	equivalent
`operator`	`op`	
`from`	`ll`	
`to`	`up`	
`of`	`opd`	
`sum`	`sum`	equivalent
`integral`	`in`	
`product`	`pr`	
Piles		
`pile`	`stk`	equivalent
`above1`	`lyr`	the role of `above1` (ISO) is not clear
`above`		
`pile` `align=(left|right|center)`	`align=(l|r|c)`	equivalent
Matrices		
`matrix`	`ar`	ISO marks up rows
`col`	`arc`	inside columns via
`above`	`arr`	above. AAP has elements
		for both.
	`ar cs`	AAP allows various
	`ar rs`	column and row
		separators.
	`ar ca`	AAP allows overall
		column alignment
`col` `align=(left|right|center)`	`arc` `align=(l|r|c|d|e)`	d and e for alignment on exponents and decimal points
Square root and square		
`sqrt`	`rad, rdx 2`	no separate elements
`square`	`sup 2`	for square roots and squares in AAP
Root and power		
`root`	`rad`	The content of `root`
`degree`	`radix`	is the content of `rad`
`of`		in AAP;
		`of` element not required.

`power`	`sup`	No general power
`degree`		element in AAP
`of`		
Open and close brackets, fences and posts		
`fence`	`fen`	equivalent
`fence style=`	`fen style=`	
`fence open=`	`fen lp=`	
`fence type=`	`fen post=`	
`fence close=`	`rp`	
	`rp style=`	
	`rp post=`	
`middle`	`cp`	
`middle style=`	`cp style=`	
	`cp post=`	

B.2.1 Short references

Note: short reference maps are defined for fences in both applications. In addition, AAP has maps for starting rows and columns in arrays, and for certain accented characters.

B.3 Facilities in AAP not available in ISO

B.3.1 Phrases in formulas

The `phr` element changes to roman font inside a formula. The ISO application uses the roman element for this.

B.3.2 Type style tags

By introducing elements that represent type styles (e.g. Greek, bold, sans serif), some redundancy is built into the AAP dtd. They are: bold (`b`), bold German (`bge`), bold Greek (`bg`), bold italic (`bi`), bold italic sans serif (`bsf`), bold script (`bsc`), Greek (`g`), italic (`it`), italic sans serif (`isf`), monospace (`ty`), openface (`op` – note that this element already exists for operator), roman (`rm`), sans serif (`ssf`), script (`sc`), and small capitals (`scp`). It is our opinion that one should differentiate between alphabets and presentation properties:

- Alphabets are: Latin, Greek, Hebrew and Cyrillic (sufficient for math)
  Properties are: roman, bold, italic, slant, sans-serif, script, fraktur, openface,. . .
- All, or most, properties can be applied to the Latin alphabet, as transformations.

Cyrillic is not a transformation of Latin, wheras bold Latin is! Some combinations of properties are also allowed, but there is, for example, no sans-serif fraktur or fraktur Cyrillic, so some combinations are meaningless.

B.3.3 Some maths objects

AAP has tags for dyadics (`dy`), and fields (`fi`).

B.3.4 Horizontal and vertical spacing

AAP has tags for horizontal space (`hsp`) and vertical space (`vsp`), for use in cases where space needs to be explicitly indicated by the author.

B.3.5 Atom change tags

Mathematical formulae are composed of atoms. There are 7 types, adapted from TEX [20]: Ord (ordinary), Op (operators), Bin (binary operation, Rel (relation), Open, Close, and Punct (punctuation). The `ach` (atom change) tag changes a character's atom type.

B.4 Future developments

There is a large overlap between the ISO and AAP (and hence Euromath) dtd's. It should be possible to make a single dtd that contains 'the best of both'. Indeed, a working group is now studying this problem under the auspices of the AAP, and a dtd designed along the lines sketched in this article is under development.

References

[1] Charles Goldfarb. *The SGML Handbook*. Oxford University Press, Oxford, 1990.

[2] Standard for electronic manuscript preparation and markup version 2.0. Technical report, EPSIG, Dublin (Ohio), 1987. ANSI/NISO Z39.59-1988.

[3] Techniques for using SGML. ISO Technical Report 9573, 1988.

[4] American Chemical Society. ACS journal dtd. (unpublished draft version).

[5] Björn von Sydow. On the `math` type in Euromath. (preprint).

[6] N.A.F.M. Poppelier. SGML and TEX in scientific publishing. *TUGboat*, 12:105–109, 1991.

[7] E. van Herwijnen, N.A.F.M. Poppelier, and J.C. Sens. Using the electronic manuscript standard for document conversion. *EPSIG News*, 1:14, 1992.

[8] E. van Herwijnen. The use of text interchange standards for submitting physics articles to journals. *Comp. Phys. Comm.*, 57:244–250, 1989.

[9] E. van Herwijnen and J.C. Sens. Streamlining publishing procedures. *Europhysics News*, pages 171–174, November 1989.
[10] International standard ISO 8879: Standard generalized markup language (SGML). Technical report, ISO, Geneva, 1986.
[11] M. Abramovitz and I. Stegun. *Handbook of mathematical functions*. Dover, New York, 1972.
[12] I.S. Gradshteyn and I.M. Ryzhik. *Tables of integrals, series, and products*. Academic Press, New York, 1980.
[13] S.A. Mamrak, C.S. O'Connell, and J. Barnes. Technical documentation for the integrated chameleon architecture. Technical report, March 1992. (Part of the documentation of the ICA software).
[14] Neil M. Soiffer. *The design of a user interface for computer algebra systems*. PhD thesis, Computer Science Division (EECS), University of California, Berkeley, 1991. Report UCB/USD 91/626.
[15] M. Nakahara. *Geometry, Topology and Physics*. Adam Hilger, Bristol, 1990.
[16] Dennis S. Arnon and Sandra A. Mamrak. On the logical structure of mathematical notation. *TUGboat*, 12:479–484, 1991.
[17] Björn von Sydow. private communication to one of us (EvH).
[18] Stephen Wolfram. *Mathematica: a system for doing mathematics by computer*. Addison-Wesley, Reading, 1991.
[19] Bruce W. Char, Keith O. Geddes, Gaston H. Gonnet, and Stephen M. Watt. *Maple User's Guide*. WATCOM Publications Ltd., 1985.
[20] Donald E. Knuth. *The TEX book*. Addison-Wesley, Reading, 1984.
[21] Joseph F. Osanna. *UNIX Programmer's Manual (2b), chapter Nroff/troff*. Bell Laboratories, 1978.
[22] Brian W. Kernighan and Linda Cherry. *UNIX Programmer's Manual (2b), chapter Typesetting mathematics*. Bell Laboratories, 1978.
[23] E. van Herwijnen. *Practical SGML*. Kluwer Academic Publishers, Dordrecht, 1990.
[24] SGML'90 report. *EPSIG News*, 3(4):4, 1990.
[25] International standard ISO 9541: Font information interchange. Technical report, ISO, Geneva, 1991.
[26] Information processing – SGML support facilities – techniques for using SGML – part 13. Proposed Draft Technical Report ISO 9573, 1991.
[27] Leslie Lamport. *LATEX: a document preparation system*. Addison-Wesley, Reading, 1985.
[28] Markup of mathematical formulas. Technical report, EPSIG, Dublin (Ohio), 1989.
[29] Information processing – SGML support facilities – techniques for using SGML – part 11. Proposed Draft Technical Report 9573, 1991.
[30] Grif SGML editor 2.1, December 1991.
[31] Yvonne Choquet-Bruhat and Cécile DeWitt-Morette. *Analysis, manifolds and physics*. North-Holland, Amsterdam, revised edition, 1987.

Incorporating PostScript fonts in TEX*

Erik-Jan Vens

Rijks Universiteit Groningen
ICCE
P.O.Box 335
9700 AH Groningen
the Netherlands
E.J.Vens@icce.rug.nl

Abstract

pfb2mf provides the TEX community with an interface to the PostScript Type One fonts. There is an overwhelming amount of these fonts for sale and there are a lot of fonts in the Public Domain, so it extends the range of typefaces the TEX user can choose from.

1 The typical look of a TEX-ed document

1.1 Some of my background

As a novice TEX[1] user, one will often be very enthousiastic about the program. At least, from those days when I was a rookie TEX user, I remember a sort of passion for what was possible with this incredible program. It was in 1985 and our university's computing centre had a Canon BLP-8 hooked up to their VAX 8650. I was a student then and used to work there almost every evening, sometimes nights. There was this typical small group of people working in the only room which was open to general users. And typically we would gather around the coffee-machine and discuss all sorts of things.

One of us knew about this program called TEX. He showed me some results and I was so impressed that the next day I visited the library to pick up 'The TEXbook'. (Needless to say, of course it was 'TEX and METAFONT: New Directions in Typesetting', so it was of little help.) Anyways, I spent many hours trying to create masterpieces of the publishing art. I was very enthousiastic about the program.

And I became even more enthousiastic when I met fellow TEXers, or saw publications made with TEX. But, how did I see these were made TEX? It was precisely because they were all typeset with the Computer Modern font family.

1.2 How and why did Knuth create the Computer Modern font family

When Knuth decided to write TEX, he wanted to be able to finish his series 'The Art of Computer Programming' with digital typography. As he writes in 'The Errors of TEX': 'The genesis of TEX probably took place on 1 February 1977, when I first chanced to see the output of a high-resolution typesetting machine. I was told that this fine typography [. . .] was produced by entirely digital methods; yet I could see no difference between the digital type and 'real' type. Therefore I realized that a central aspect of printing had been reduced to bit manipulation. As a computer scientist, I could not resist the challenge to improve print quality by manipulating those bits better. [. . .] By 13 February I had changed my plan to spend a forthcoming sabbatical year in South America; instead of travelling to an exotic place and working on Volume 4 of *The Art of Computer Programming*, I had decided to stay at Stanford and work on digital typography.'

He also needed a typeface. He chose Monotype Modern No. 8A, of which he says in 'Computers & Typesetting, Vol. E': 'In letterpress printing, modern fonts were technically troublesome because their delicate hairlines and serifs were particularly susceptible to damage; the broken letters produced a degraded text image. In digital typography with sufficient resolution, the delicate forms of modern can be rendered precisely. Computer typesetting and photolithographic printing are capable of reproducing modern typefaces with a clarity and sharpness unobtainable at the time of the original development of the style'.

Undoubtly, this is true. Anyone who has ever seen a text set in Computer Modern come out of a phototypesetter, knows this striking feeling of beauty. Only, most applications of TEX that I have seen, do not end up on a 2400 dpi phototypesetter. The typical output resolution is 300 dpi for a laser printer, 240 dpi for a dotmatrix

*Presented at EuroTEX '92, September 14–18, Prague, Czechoslovakia.

[1] Just as some people do mention that they denote 'he and/or she' by just using 'he', my usage of 'TEX' means 'TEX and/or LATEX'.

printer and 400 dpi for the NeXT laser printer. So choosing fonts that render well at low resolutions is certainly worthwhile.

1.3 Why there are no serious attempts at creating other faces

Creating a typeface from scratch is an extremely difficult task that takes lots of time, patience and may be energy. The only other complete family of typefaces designed with METAFONT is Neenie Billawala's Pandora. This is not just a recreation of an existing font, it is a series of tools, drivers and parameters that make up a family of faces, as she has described in TUGboat, Vol. 10, (1989), No. 4, pp. 481–489.

I am still working on a font together with a friend, where I do most of the programming and he does most of the designing. But this is our long, cold winter's sunday afternoon project, so its not nearly finished, even though we started out one-and-a-half-year ago.

And there are more people who have done some work in this area. Yannis Haralambous did some. He and other people have created new fonts, notably for Indian languages, Greek, Hebrew and Russian. But the TEX community is not a very large one, and so the field of font creation remains a largely undiscovered one.

2 The wealth of the world of PostScript fonts

But may be we do not need to wait for new fonts written in METAFONT. One of the many attractions of the PostScript world is its large range of available typefaces. Even in the Public Domain one can find numerous nice looking fonts. And if you consider buying them, there is such a large choice, that it becomes virtually impossible to get to know them all before taking your pick. So one tends to start at the traditional, standard PostScript look of a document.

2.1 The standard Times Roman plus Courier document

In much the same way that a TEXed document can be recognized by the use of Computer Modern, can a standard PostScript document be recognized by the combination of Times Roman and Courier, with Courier being slightly too thin next to the Times. Most people who come from the world of computer science know this combination from books as Kernighan & Ritchie's 'The C Programming Language' and 'Kernighan & Pike's 'The UNIX Programming Environment'.

Other traditional PostScript fonts include AvantGarde, Bookman, Helvetica, New Century Schoolbook, Palatino and some loose fonts as Zapf Dingbats and Zapf Chancery.

2.2 But there are more fonts

The past few years –as fonts grew cheaper and cheaper– many users have got used to be able to buy fonts. If one looks at the Top 40 bestsellers list of Adobe, one will notice that both in Europa and in the USA, Adobe Garamond is in the Top 10. On the other hand, Neville Brody's Arcadia, Industria and Insignia are the number one in the USA, where in Europe they rank only at 27. In Europe e.g. the GillSans ranks very high. Even in the 'Jobs' section of a widely read Dutch newspaper one can find many, many uses of GillSans. Even Flora and Praxis (both by Dutch designer Gerard Unger) can be found often.

3 Combining the worlds of TEX and PostScript fonts

Wouldn't it be nice if the user of TEX could incorporate the use of PostScript fonts in a TEX document? TEX still is one of the best, if not **the best** where the production of neatly spaced text and beautifully typeset mathematics is concerned. We –as TEX users– shiver in awe when we are confronted with the Swiss cheese output produced by WorstPervert, let alone the `eqn`-based formula editor it now provides. We don't even want to think about WordStar, and we have seen instances where the output of Ventura doesn't look extremely bad. We have heard that Word4Windows is nice. But still, TEX outranks them when pure text and mathematics is concerned.

3.1 Choosing a scheme

There are several possibilities open to the non Computer Modern user.

1. Completely switch to PostScript output. There is a Public Domain PostScript emulator, written by L. Peter Deutsch, in the GNU distribution. One will also have to use a `dvi` to `ps` translator, e.g. Tom Rokicki's `dvips`.
2. Use Piet Tutelaer's `ps2pk` package, based on the X-Windows, Release 5 distribution, which can translate a PostScript Type One font to TEX's `pk` format. This is extremely useful for those users who want fast, production quality fonts.
3. Use my `pfb2mf` package, which can translate a PostScript Type One font to METAFONT format. To create a bitmap, one will still have to run METAFONT. This last part of the job can be frustratingly slow, but it does provide the TEX user with options to change the output.

Beware. A caveat should be here. Producing beautifully typeset documents is just combining spaces and faces. But remember that this is an art, rather than a job which can be automated.

4 What does a METAFONT file look like

A METAFONT file consists of a combination of commands to set up the communication with TEX, commands to set up plottable functions and commands to set up a plot.

4.1 The macro-biotic aspects of METAFONT

Just as with TEX, macros play an important role. And just as with TEX, macros are essentially built up from other macros and primitives. Only because with TEX one has to make a distinction between pure text and macros, the macros have to be escaped with a backslash. METAFONT macros on the other hand do not need the escape. And METAFONT macros can even handle binary operators. An example on p. 178 of 'The METAFONT book' states:

primarydef w dotprod z =

(xpart$w$ * xpart$z$ + ypart$w$ * ypart$z$) **enddef**.

I will not discuss this in in-depth, but basically this defines an infix operator '`dotprod`' which takes a left and a right operand, it then returns a value.

There a couple of important macros and primitives:

- '`beginchar`' sets up the beginning of the definition of a glyph. It takes four parameters:
  1. The character number, this can be given as a decimal, octal or hexadecimal number, or as the representation of the character, e.g. `"a"`.
  2. The width of the surrounding box. The character's shape can stick out of its box, but this box provides the metrics with which TEX will do its calculations.
  3. The height of the surrounding box. This is the height the characters extends above the baseline.
  4. The depth of the surrounding box.
- '`endchar`' declares the end of the definition.
- One can assign values to constants and variables. $u := 3.5$, or $width\# := 3.5pt\#$, where the $\#$ denotes a sharped, i.e. reallife, or not-rounded-to-the-raster value.
- One can declare points in space, or just their x- and y-values: $x_3 = x_2 - x_1$, or $z_1 = (\frac{1}{2}width, .8h - 3.5pt)$.
- The points can be connected with several types of lines. E.g. a straight line: $z_1 -- z_2 -- z_3$, or a smooth curve: $z_1 .. z_2 .. z_3$.
- One can draw these lines with a pen, or one can fill the area that is defined by a shape. Or combining these two. One can also erase a line, or an area. `draw` $z_1 -- z_2$ draws a straight line from z_1 to z_2 with the current pen. `fill` $z_1 .. z_2 -- z_3 .. cycle$ fills an area defined by a smooth line from z_1 to z_2, from there with a straight line to z_3, and from there with a smooth line back to the beginning.

5 What does a Type One font look like

We will have to do some work before we can take a look at Type One fonts with a plain text editor.

5.1 From `pfb` via `pfa` to `ps`

Type One fonts typically come in two flavors: binary or ascii representation. The binary representation is just an encoded form of the ascii representation. So, one needs a program to translate from the first to the last form. In the `pfb2mf` package that would be `pfb2pfa`. The ascii version, however, is a readable hexadecimal version of an encrypted series of PostScript commands. Once this encryption was secret, but it is now given to the world. So, one can translate the ascii version to human-readable PostScript. In the `pfb2mf` package this would be done with `pfa2ps`.

5.2 What does it mean

All Type One fonts are outline fonts which are filled. The PostScript language uses a reverse polish notation for its commands, so '`dx dy rlineto`' means 'draw a straight line from the current point to the point which lies at a distance (dx, dy). There are very few commands:

- '`endchar`' finishes a charstring outline definition.
- '`seac`' makes an accented character from two other characters in the font program.
- '`closepath`' closes a subpath. (Surprise!)
- All sorts of drawing and moving commands. These are all relative to the current point. Some commands are abbreviations for others, one can say, e.g. `dx hmoveto`, which means `dx 0 rmoveto`. All lines are straight lines. The curves are Bézier cubics, so `dx1 dy1 dx2 dy2 dx3 dy3 rrcurveto` means 'construct a curve from $currentpoint$ to $currentpoint + (dx1 + dx2 + dx3, dy1 + dy2 + dy3)$, using $currentpoint + (dx1, dy1)$ and $currentpoint + (dx1 + dx2, dy1 + dy2)$ as control points.
- Then there are the stem commands. This is the most difficult to understand part of a Type One font. I am still not sure whether I really do. Anyways, stem commands define zones within which the renderer can choose a position so that the output will look good. It is strongly related to METAFONT's commands for 'Discreteness and Discretion'.
- We can define subroutines and call them from within the Type One font program. So, any translation program must learn these subroutines and apply them when called.
- '`setcurrentpoint`' sets the current point, for how else can we start a curve where each point is defined in terms of its predecessor?

6 The uninteresting bits and pieces

The definition of how this is all done is not very interesting, so I will give just one example and leave it to the reader to pick up the sources if this reader is interested

in more details.

6.1 An example of the font

There is a freely distributable version of Times New Roman out on the net in TrueType format. I have translated this to Type One format. This is an example of what the fonts look like:

Times New Roman

6.2 An example of the letter A in sourceform

```
/A {7 722 hsbw 450 221 rmoveto -256
    hlineto -45 -104 rlineto -12 -26 -5
    -20 0 -12 rrcurveto 0 -10 5 -9 10 -8
    rrcurveto 9 -7 21 -5 32 -2 rrcurveto
    -18 vlineto -209 hlineto 18 vlineto
    28 4 18 7 8 8 rrcurveto 16 15 19 32
    20 48 rrcurveto 233 545 rlineto 17
    hlineto 231 -551 rlineto 18 -44 17
    -29 15 -13 rrcurveto 15 -13 21 -8 27
    -1 rrcurveto -18 vlineto -261 hlineto
    18 vlineto 27 1 17 4 10 8 rrcurveto 9
    7 5 9 0 11 rrcurveto 0 14 -7 23 -13
    31 rrcurveto -40 95 rlineto -14 36
    rmoveto -112 268 rlineto -115 -268
    rlineto 227 hlineto endchar } ND


beginchar(65,722*FX#,677*FY#,0*FY#); "A";
fill ((457,221) -- (201,221) -- (156,117)
     .. controls (144,91) and (139,71) ..
     (139,59) .. controls (139,49) and
     (144,40) .. (154,32) .. controls
     (163,25) and (184,20) .. (216,18) --
     (216,0) -- (7,0) -- (7,18) ..
     controls (35,22) and (53,29) ..
     (61,37) .. controls (77,52) and
     (96,84) .. (116,132) -- (349,677) --
     (366,677) -- (597,126) .. controls
     (615,82) and (632,53) .. (647,40) ..
     controls (662,27) and (683,19) ..
     (710,18) -- (710,0) -- (449,0) --
     (449,18) .. controls (476,19) and
     (493,23) .. (503,31) .. controls
     (512,38) and (517,47) .. (517,58) ..
     controls (517,72) and (510,95) ..
     (497,126) -- (457,221) -- cycle)
     xscaled FX# yscaled FY#;
unfill ((443,257) -- (331,525) -- (216,257)
     -- (443,257) -- cycle) xscaled FX#
     yscaled FY#;
endchar;
```

7 Does it meet realworld constraints?

No, as of yet, it doesn't. However, the process of hinting is time-consuming and therefor very, very slow. So, if it is fast production quality you are after, you are well advised to pick up Piet Tutelaer's `ps2pk`.

I want to extend the possibilities of the program to provide the user with ways of implementing meta-ness. So if you'd like to fiddle with the font's parameters, `pfb2mf` is what you want. It will provide a great deal of tuneable parameters, and you can lift out part of a font an use that as a basis for other things.

8 And where can I get all this?

Both `pfb2mf` and `ps2pk` are ftp-able. E.g. from my home machine `obelix.icce.rug.nl`. It can be found in `pub/erikjan`.

9 A warning

Be careful! If you plan on using more fonts, you will not only have to worry about your text, and the layout you choose, you will also have to worry about what font you are going to use.

Creating Shaded Rectangles with PostScript

David Salomon

11835 Sapota Dr.,
Lakeside, CA 92040, USA
dxs@ms.secs.csun.edu

June 1992

Abstract

One of the most common graphics used in documents is text with a shaded background. This is hard to do with TEX but easy with PostScript. Simple PostScript code is presented here to create shaded rectangles, and a macro is developed to combine such a rectangle with text.

1 Introduction

It is well known that TEX lacks facilities for graphics. Certain drawings, such as logos, may be developed in METAFONT, but for any non-trivial graphics this is usually too time consuming. A ruled box, like this, is the closest TEX can get to anything resembling a drawing. Slanted lines can be handled by typesetting a dot and moving it in small steps. Hendrickson (1985), Cameron (1985) and Salomon (1989) use this idea to develop simple macros for slanted lines. The same principle can be used to typeset curves, as done by `Bezier.sty` in LATEX, and by the PICTEX macro package (Wichura, 1986). For more complex graphics, the `\special` command can be used to include graphics in the final document. Rahtz (1989) is an excellent survey of the different methods used to combine TEX and graphics.

In my work I have often felt the need for enclosing text in ruled boxes, either rectangular or with rounded corners, with backgrounds of shaded gray, and sometimes with a thick stroke around them. Glendown (1989) is an (unsatisfactory) attempt to draw similar boxes using the arcs provided in the LATEX circle fonts. As a heavy Macintosh user, it seemed to me that my best choice was to use Postscript to achieve such effects. Postscript printers are widely available, and Postscript programs are supported, via `\special`, by more and more TEX implementations. The program used here has the additional advantage of being small and fast.

2 The `\shade` Macro

The result of my efforts is the macro `\shade` below. It creates a rectangular box with either sharp or rounded corners around text. The text can be placed, by the user, in either an `\hbox` or a `\vbox`, or it can be written explicitly as the macro argument (see third example below). The rectangle is filled with the desired shade of gray, and is optionally surrounded by a stroke of any desired thickness. It is also possible to make the shaded area larger than the text by any amount. The macro has five parameters:

- A decimal number specifying the proportion of white in the shaded background.
  This is a .97 background.
- A dimension specifying the
  extra size of shaded area (12pt on each side).
- A count specifying the width of the stroke in points. A zero or empty argument creates no stroke.
- A count—the radius of the rounded corners, in points. A zero or empty argument produces square corners.
- The text to be boxed. It is either straight text (if it fits on one line) or is enclosed, by the user, in an `\hbox` or `\vbox`.

The arguments are separated by commas, and the last one is delimited by '`\\`'.

Here is the listing of `\shade`.

```
\newdimen\tmp \newdimen\tmpw
\newdimen\tmpk \newdimen\tmph
\newcount\heit \newcount\widf
\newcount\strk \newcount\radius
\def\shade#1,#2,#3,#4,#5\\{%
\setbox1=\hbox{#5}%
\def\inner{#3}\ifx\inner\empty \strk=0
 \else\strk=#3\fi
\def\inner{#4}\ifx\inner\empty \radius=0
\else \radius=#4\fi
\tmp=#2 \tmp=2\tmp \advance\tmp\wd1
 \widf=\tmp \divide\widf65536
\tmp=#2 \tmp=2\tmp \advance\tmp\ht1
```

```
 \heit=\tmp \divide\heit65536
\setshadebox{#1}{\the\heit}{\the\widf}
 {\the\strk}{\the\radius}%
\tmp=#2
\tmpw=#2 \tmpw=2\tmpw
 \advance\tmpw by\the\strk pt
 \advance\tmpw\wd1
\tmpk=\the\strk pt \divide\tmpk by2
\tmph=#2 \tmph=2\tmph
 \advance\tmph by\the\strk pt
 \advance\tmph\ht1
\vbox to\tmph{\vss
\hbox to\tmpw{\kern\tmpk\lower\tmp\box0%
 \kern-\tmpk\hfil\box1\hfil}\vss}}
% The ps commands are placed, as the
% argument of a \special, at the
% bottom of \box0 whose width is set
% to zero.
\def\setshadebox#1#2#3#4#5{%
\setbox0=\vbox to\tmp{\hsize=0pt\vfil
 \special{postscript
/fpops{4 {pop} repeat} def
#3 0 moveto 0 0 0 #2 #5 arcto
newpath % start the rounded-corner
        % rectangle
moveto % the coordinates are in stack
0 #2 #3 #2 #5 arcto fpops
#3 #2 #3 0 #5 arcto fpops
#3 0 0 0 #5 arcto fpops
0 0 0 #2 #5 arcto fpops closepath
#4 0 ne
 {gsave #4 setlinewidth stroke
  grestore} if
% stroke width of zero not recommended
#1 setgray fill}}}
```

The `\special` with the Postscript commands is placed at the bottom of `\box0`, which is set to width zero. The text is placed in `\box1`. Both boxes are then placed, side by side, in an `\hbox` set to the right width (the width of the text, plus the extra width of the shaded area and the stroke).

The macro is normally used in vertical mode, but also works well in horizontal mode, not introducing any spurious spaces (the percent signs at the end of certain lines should not be removed). It has also been tested in both math modes. Note that Postscript code should normally be surrounded by a `gsave`, `grestore` pair. However, I use TEXtures, which automatically supplies these commands.

3 Examples

The following examples show samples of input and their respective output:

```
\shade0.90,16pt,5,10,\vbox{\hsize=100pt
\noindent A test in vertical mode.
Filled with 90\% white \&
a 5pt stroke}\\
```

A test in vertical mode. Filled with 90% white & a 5pt stroke

```
\shade0.97,5pt,1,3,%
straight text, not pre-boxed\\
```

straight text, not pre-boxed

```
\shade0.90,5pt,,3,%
\hbox{No stroke at all}\\
```

No stroke at all

```
\noindent A\lower30pt\hbox{%
\shade0.98,10pt,1,5,
\vbox{\hsize=100pt
 A test in horizontal mode. There are
no spaces between this text and the
‘A’ or the ‘D’ (98\% white)}\\}D
```

A test in horizontal mode. There are no spaces between this text and the ‘A’ or the ‘D’ (98% white)

D

```
\shade0.99,8pt,40,,%
\hbox{A very heavy stroke}\\
```

A very heavy stroke

```
\shade0.5,5pt,1,5,\hbox{\shade.99,5pt,1,3,%
 A Nested Expansion\\}\\
```

A Nested Expansion

4 Using the Macro in Practice

In practice, one usually uses shaded boxes of the same type throughout a document. This suggests using macros such as

```
\def\shadedbox#1{\shade0.95,1em,1,,#1\\}
\def\shadedRbox#1{\shade0.95,1em,1,5,#1\\}
```

instead of using of `\shade` directly.

The following examples below combine `\shade` with rules and leaders.

```
\vbox{\offinterlineskip
\shade0.97,4pt,1,,Section 3:\\
\hrule height4pt}
```

Section 3:

```
\hbox{\vrule width4pt%
 \shade0.95,4pt,,,Note!\\}
```

Note!

```
\line{\leaders\hbox{%
 \shade0.9,0pt,1,,\vrule height4pt width0pt%
 \kern6pt\\\kern4pt}\hfil}
```

```
\line{\leaders\hbox{%
 \shade0.9,0pt,1,2,\vrule height4pt width0pt%
 \kern6pt\\\kern4pt}\hfil}
```

References

[1] Cameron, A.G.W. “Wiggly Lines.” *TUGboat* **6**(3), pages 155–156, 1985.

[2] Glendown, G. “Rounded Boxes for `plain` TEX.” *TUGboat* **10**(3), pages 385–386, 1989.

[3] Hendrickson, A. “Some Diagonal Line Hacks.” TUGboat **6**(2), pages 83–86, July 1985.

[4] Rahtz, S. *A Survey of TEX and Graphics.* University of Southampton, Tech. Rep. CSTR 89-7, 1989.

[5] Salomon, D. “DDA Methods in TEX.” *TUGboat* **10**(2), pages 207–216, 1989.

[6] Wichura, M.J. *The PICTEX Manual.* Number 6 in the *TEXniques* series. Providence: TEX Users Group, 1986.

Introduction to MetaPost*

John D. Hobby

AT&T Bell Laboratories
600 Mountain Ave.
Murray Hill, New Jersey 07974
hobby@research.att.com

Abstract

MetaPost is a picture-drawing language very much like METAFONT except with PostScript output. The language provides access to all major features of Level 1 PostScript® and it has facilities for integrating graphics with typeset text.
This paper gives a brief overview of the MetaPost language and how it can be used. A few of the more interesting features are described in detail.

Keywords: Metapost, graphics languages, METAFONT, PostScript

1 Introduction

Although METAFONT was originally designed as a font-making tool, many people have recognized that it is also a powerful graphics language. The problem is that METAFONT's output is in the form of bitmap images instead of graphics primitives. A diagram can sometimes be created in METAFONT and typeset as a single huge character, but this is cumbersome and makes it difficult to deal with textual labels. A good examples of work along these lines appears in [4] and [8].

Another approach is to modify the METAFONT interpreter so that it outputs PostScript. Previous work along these lines presented in [1] and [10] has concentrated on producing PostScript fonts rather than graphics. Unlike these earlier systems, the MetaPost system involves the creation of a new language similar to METAFONT, but specifically designed for producing PostScript graphics. Preliminary comments on MetaPost appeared in [2].

Since MetaPost is based on the public-domain METAFONT source code given in [6], MetaPost has been able to inherit all the features of METAFONT that make it a powerful graphics language:

- The ability to store and manipulate coordinate pairs, straight and curved paths, coordinate transformations, pen shapes, and complete pictures.
- A Flexible and powerful mechanisms for constructing smooth curves and straight lines.
- The ability to draw straight and curved lines of any thickness and to fill a region given its boundary.
- Mechanisms for solving linear equations so that geometric information can be specified in a largely declarative manner.
- A powerful macro facility that allows the language to be extended syntactically and semantically.
- Operators for intersecting curves, finding tangent lines, finding points on a curve that match a given tangent direction, and extracting subpaths.

In addition to these features, MetaPost allows pictures to contain text, dashed lines, clipping paths, and areas filled with gray or other colors. There are also data types for colors and recipes for dashed lines. In addition, there are important facilities for generating and manipulating typeset text. Readers familiar with other graphics languages such as Kernighan's *Pic* [5] and Wichura's PicTEX [9] will see that MetaPost is considerably more powerful.

Section 2 gives a general idea of what the language is like and what can be done with it. More detailed discussions of interesting features follow in Section 3. This includes Section 3.1 on integrating text and graphics, Section 3.2 on dealing with dashed lines, and Section 3.3 on drawing arrows. Finally, Section 4 deals with macro packages and Section 5 presents some concluding remarks.

2 Overview of the Language

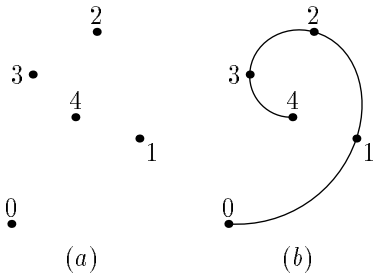
MetaPost is a batch-oriented graphics language that achieves great power and flexibility by giving up some the ease of use found in interactive graphics editors such as MacDraw. A MetaPost user prepares an input file such as the one shown in Figure 1. Invoking the MetaPost interpreter produces an encapsulated PostScript output file that can be included in a TEX doc-

*Presented at EuroTEX '92, September 14–18, Prague, Czechoslovakia.

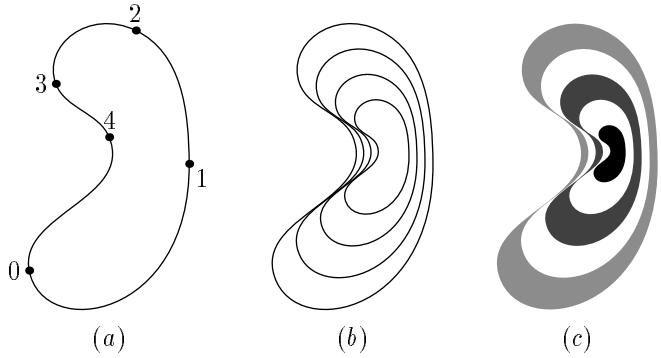
```
beginfig(1);
draw (20,20)--(0,0)--(0,30)--(30,0)--(0,0);
endfig;

end
```

Figuur 1: *A MetaPost input file and the resulting output*



Figuur 2: *A sequence of points and a curve formed by connecting them.*



Figuur 3: *A closed curve and some effects that can be achieved by rescaling it.*

ument or viewed with a PostScript interpreter such as GhostScript. The input file in the figure has a single `beginfig...endfig` block. There could be more such blocks, in which case each would produce a separate output file

Since this paper is not intended to be a user's manual, no attempt will be made to show the exact syntax used to create subsequent examples. Instead, we concentrate on general concepts with the aim of showing what MetaPost can do. The interested reader can refer to [3] for details.

Another thing MetaPost can do is draw curved lines. If points P_0, P_1, ..., P_4 are as in Figure 2a, asking the interpreter to connect them in order produces the curve in Figure 2b.

Asking for a smooth closed curve through the same sequence of points produces Figure 3a. Since MetaPost has data types and operators for objects like curved lines, it is possible to store the curve in a *path variable* `p` and use a statement like

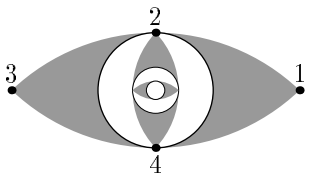
```
draw p scaled s
```

to draw rescaled versions of `p`. Figure 3b was generated by placing this statement in a loop that scans various values of `s`.

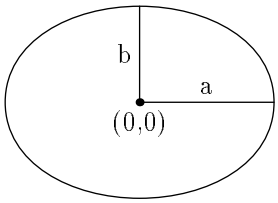
There is also a *fill* statement that fills the interior of a closed curve with a color or a shade of gray. The filled regions in Figure 3c illustrate how overlapping fills overwrite each other. The figure was generated by filling the outermost curve with light gray, then filling the next smaller curve with white, then the next smaller curve with dark gray, etc.

The examples given so far suggest that MetaPost allows drawing and filling, it has data types for numbers, coordinate pairs, and curved paths, it has operators for doing things like rescaling paths, and it has programming-language constructions such as loops. It also inherits from METAFONT the ability to solve linear equations and deal with a broad class of coordinate transformations.

Figure 4 illustrates linear equations and coordinate transformations. It was generated by introducing an unknown transformation `T`, giving a pair of equations



Figuur 4: *An example of repeated transformations.*



Figuur 5: *A labeled diagram.*

that declare it to be shape-preserving, and declaring that it maps Point 1 into Point 2 and Point 3 into Point 4. The figure was created by generating a simple picture `P` and repeatedly drawing `P` and transforming it by `T`.

3 Interesting Features

Anyone familiar with METAFONT can see that Section 2 did not begin to cover all the language features mentioned in the introduction. While it is impractical to give a detailed treatment of the entire language, we can concentrate on a few of the features that distinguish MetaPost from METAFONT and from other graphics languages.

3.1 Text in Pictures

MetaPost has a number of features for including labels and other text in the figures it generates. The simplest way to do this is to use the `label` statement to specify the label text and the point to be labeled. If you are labeling some feature of a diagram you probably want to offset the label slightly to avoid overlapping. This is illustrated in Figure 5 where statements of the form

`label.top("a",` ⟨expression$_1$⟩`);`
`label.lft("b",` ⟨expression$_2$⟩`);`

put the `"a"` label above the midpoint of the line it refers to and the `"b"` label is to the left of the midpoint of its line. (In addition to `top` and `lft`, there are six other optional suffixes for other label positions.)

There is also a `dotlabel` command that marks a point with a dot and positions text as the `label` command does. For instance, the command

```
dotlabel.bot("(0,0)", (0,0))
```

generates a dot marked “(0,0)” as in Figure 5.

For labeling statements such as `label` and `dotlabel` that use a string expression for the label text, the string gets typeset in a default font as determined by the string variable `defaultfont`. The initial value of `defaultfont` is likely to be `"cmr10"`, but it can be changed to a different font name by giving an assignment such as

```
defaultfont:="Times-Roman"
```

When you change `defaultfont`, the new font name should be something that TEX would understand since MetaPost gets height and width information by reading the `tfm` file. (See [7]). It should be possible to use built-in PostScript fonts, but the names for them are system-dependent. Some systems may use `rptmr` or `ps-times-roman` instead of `Times-Roman`. A TEX font such as `cmr10` is a little dangerous because it does not have a space character or certain ASCII symbols. In addition, MetaPost does not use the ligatures and kerning information that comes with a TEX font.

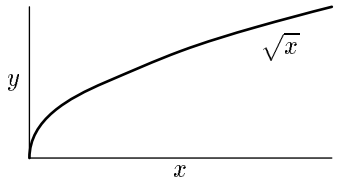
The MetaPost language does not need elaborate typesetting abilities because there is a preprocessor that extracts TEX commands, runs them through TEX (or LaTEX), and translates the output into a form that the interpreter understands. There is even a separate preprocessor that handles troff commands. Any time you say

`btex` ⟨typesetting commands⟩ `etex`

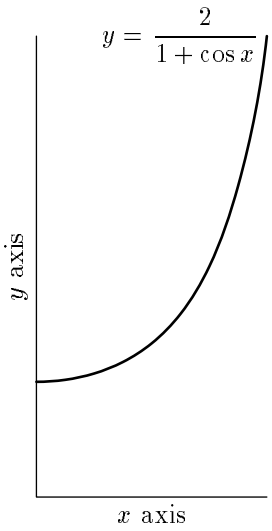
in a MetaPost input file, the preprocessor translates the ⟨typesetting commands⟩ into a MetaPost picture expression that can be used in a `label` or `dotlabel` statement. For instance, a statement of the form

`label.lrt(btex $\sqrt x$ etex,` ⟨coordinates⟩`)`

was used to placed the label $\sqrt{x}$ in Figure 6.



Figuur 6: *A figure with labels done in TEX.*



Figuur 7: *An example of how typeset labels can be rotated*

Figure 7 illustrates some of the more complicated things that can be done with labels. Since the result of `btex ... etex` is a picture, it can be operated on like a picture. In particular, it is possible to rotate the picture by giving

```
btex $y$ axis etex rotated 90
```

as the argument to a `label` statement.

Here is how TEX material gets translated into a form MetaPost understands: The MetaPost processor skips over `btex ... etex` blocks and depends on a preprocessor to translate them into low level MetaPost commands. If the main file is `fig.mp`, the translated TEX material is placed in a file named `fig.mpx`. This is normally done silently without any user intervention but it could fail if one of the `btex ... etex` blocks contains an erroneous TEX command. Then the erroneous TEX input is saved in the file `mpxerr.tex` and the error messages appear in `mpxerr.log`.

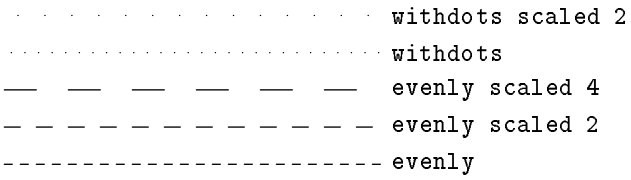
TEX macro definitions or any other auxiliary TEX commands can be enclosed in a `verbatimtex ... etex` block. The difference between `btex` and `verbatimtex` is that the former generates a picture expression while the latter only adds material for TEX to process. For instance, if you want TEX to typeset labels using macros defined in `mymac.tex`, your MetaPost input file would look something like this:

```
verbatimtex \input mymac etex
beginfig(1);
  ⋮
label(btex ⟨TEX material using mymac.tex⟩
                    etex, ⟨coordinates⟩);
  ⋮
```

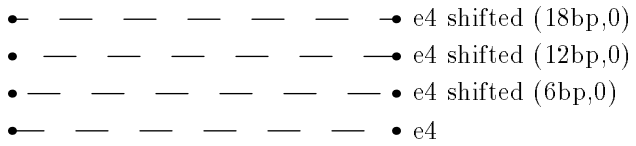
MetaPost has an internal variable called `prologues` that controls the handling of text in pictures. Giving this internal variable a positive value causes causes output to be formatted as "structured PostScript" generated on the assumption that text comes from built-in PostScript fonts. This makes MetaPost output much more portable, but it generally does not work with TEX fonts unless you have them in PostScript Type 1 format. Many `dvi`-to-PostScript programs download bitmaps for only those characters actually used in the document. Such programs can handle MetaPost output if they understand the nonstandard PostScript comments that the MetaPost interpeter uses to indicate which characters need to be downloaded. Recent versions of Rokicki's `dvips` have this capability.

3.2 Dashed Lines

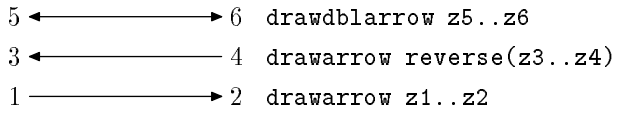
The MetaPost language provides many ways of changing the appearance of a line besides just changing its width. This is done by specifying a *dash pattern* when drawing a straight or curved line. Figure 8 shows a few examples of dash patterns and the lines they



Figuur 8: *Dashed lines each labeled with the dash pattern used to create it.*



Figuur 9: *Dashed lines each labeled with the corresponding dash pattern, where* `e4` *refers to the dash pattern* `evenly scaled 4`.



Figuur 10: *Three ways of drawing arrows.*

generate. There is a predefined dash pattern called `evenly` that makes dashes 3 points long separated by gaps of the same size. Another predefined dash pattern `withdots` produces dotted lines with dots 5 points apart. As shown in the figure, scaling the dash pattern produces dots further apart or longer dashes further apart.

Another way to change a dash pattern is to alter its phase by shifting it horizontally. Shifting to the right makes the dashes move forward along the path and shifting to the left moves them backward. Figure 9 illustrates this effect. The dash pattern can be thought of as an infinitely repeating pattern strung out along a horizontal line where the portion of the line to the right of the y axis is laid out along the path to be dashed.

When you shift a dash pattern so that the y axis crosses the middle of a dash, the first dash gets truncated. Thus the line with dash pattern `e4` starts with a dash of length 12bp followed by a 12bp gap and another 12bp dash, etc., while `e4 shifted (18bp,0)` produces a 6bp dash, a 12 bp gap, then a 12bp dash, etc. This dash pattern could be specified more directly via the `dashpattern` function:

```
dashpattern(on 6bp off 12bp on 6bp)
```

This means "draw the first 6bp of the line, then skip the next 12bp, then draw another 6bp and repeat." If the line to be dashed is more than 30bp long, the last 6bp of the first copy of the dash pattern will merge with the first 6bp of the next copy to form a dash 12bp long.

3.3 Arrows

Drawing arrows like the ones in Figure 10 is simply a matter of saying

`drawarrow` ⟨path expression⟩

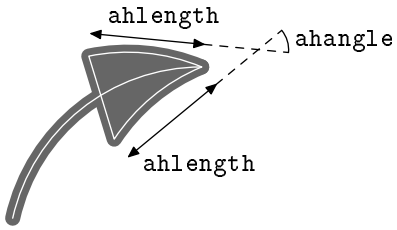
instead of `draw` ⟨path expression⟩. This draws the given path with an arrowhead at the end. If you want the arrowhead at the beginning of the path, there is an operator that reverses a path. For double-headed arrows, there is a `drawdblarrow` statement.

The size of the arrowhead is guaranteed to be larger than the line width, but it might need adjusting if the line width is very large. This is done by assigning a new value to the internal variable `ahlength` that determines arrowhead length as shown in Figure 11. Increasing `ahlength` from the default value of 4 PostScript points to 1.5 centimeters produces the large arrowhead in Figure 11. There is also an `ahangle` parameter that controls the angle at the tip of the arrowhead. The default value of this angle is 45 degrees as shown in the figure.

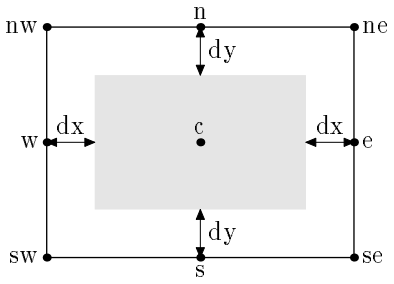
The arrowhead is created by filling the triangular region that is outlined in white in Figure 11 and then drawing the boundary with the current line width. Readers familiar with METAFONT will recognize this as the `filldraw` statement.

4 Macro Packages

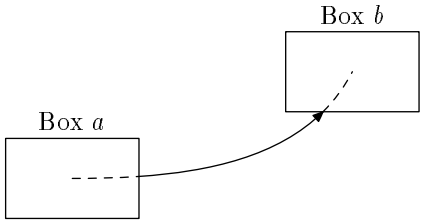
This section describes auxiliary macros not included in Plain MetaPost. The macros described in Section 4.1



Figuur 11: *A large arrowhead with key parameters labeled and paths used to draw it marked with white lines.*



Figuur 12: *The relationship between the picture given to* `boxit` *and the associated variables. The picture is indicated by a gray rectangle.*



Figuur 13: *A "well-targeted arrow" generated by trimming the dashed sections from a curved path.*

make it convenient to do things that *pic* is good at [5]. Section 4.2 makes some brief remarks about other macro packages. In order to use a macro package, it is necessary to give a MetaPost command that names the macro file and asks the interpreter to read it.

4.1 Macros for Boxes

The box-making macros are contained in a macro file called `boxes.mp`. This file can be accessed by giving the MetaPost command `input boxes` before any figures that use the box making macros.

The basic tool for making boxes is the command

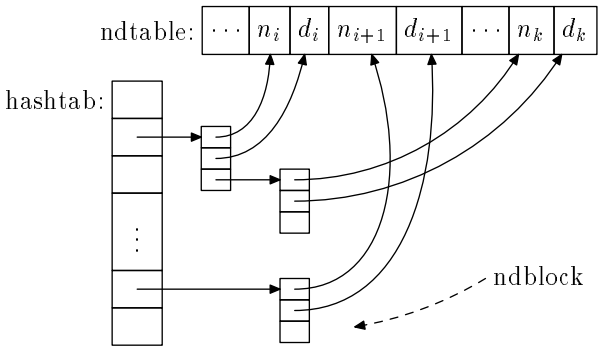
`boxit.`⟨box name⟩`(`⟨picture expression⟩`)`

This creates variables ⟨box name⟩`.c`, ⟨box name⟩`.n`, ⟨box name⟩`.e`, ... that can then be used for positioning the picture before drawing it. The actual drawing is done by a separate command `drawboxed` that takes a list of box names.

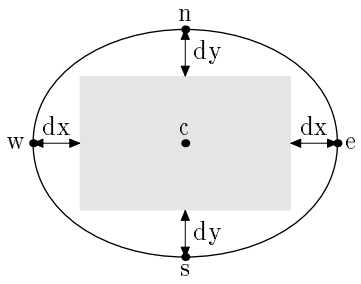
If the command is `boxit.bb(`⟨picture⟩`)`, the box name is `bb` and the contents of the box is the ⟨picture⟩. In this case, `bb.c` the position where the center of the picture is to be placed, and `bb.sw`, `bb.se`, `bb.ne`, and `bb.nw` are the corners of a rectangular path that will surround the picture. Variables `bb.dx` and `bb.dy` give the spacing between the picture and the surrounding rectangle, and `bb.off` is the amount by which the picture has to be shifted to achieve all this.

When the `boxit` macro is called with box name b, it gives linear equations that force b`.sw`, b`.se`, b`.ne`, and b`.nw` to be the corners of a rectangle aligned on the x and y axes with the box contents centered inside as indicated by the gray rectangle in Figure 12. The values of b`.dx`, b`.dy`, and b`.c` are left unspecified so that the user can give equations for positioning the boxes. If no such equations are given, macros such as `drawboxed` can detect this and give default values. The default values for `dx` and `dy` variables are controlled by the internal variables `defaultdx` and `defaultdy`.

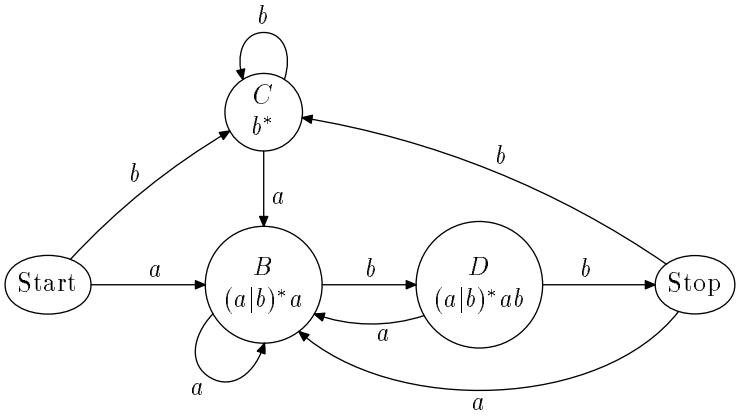
If b represents a box name, `drawboxed(`b`)` draws the rectangular boundary of box b and then the contents of the box. This bounding rectangle can be accessed



Figuur 14: *An example of what can be done with the* `boxes.mp` *macros*



Figuur 15: *The relationship between the picture given to* `circleit` *and the associated variables. The picture is indicated by a gray rectangle.*



Figuur 16: *Circular and oval boxes generated using the* `boxes.mp` *macros.*

separately as `bpath b`, or in general

`bpath` ⟨box name⟩

One interesting use of the bounding rectangle is for generating "well-targeted arrows" as shown in Figure 13. Given a path from the center of Box a to the center of Box b, there are MetaPost operators that make it convenient to chop off the parts of the path before the first intersection with `bpath a` and after the last intersection with `bpath b`.

There is also a special command

`boxjoin(`⟨equation text⟩`)`

that controls the relative position of consecutive boxes. Within the ⟨equation text⟩, `a` and `b` represent the box names given in consecutive calls to `boxit` and the ⟨equation text⟩ gives equations to control the relative sizes and positions of the boxes. For example, the MetaPost code for Figure 14 uses

```
boxjoin(a.se=b.sw; a.ne=b.nw)
```

to causes boxes to line up horizontally. (It is instructive to compare this figure with the similar one in the pic manual [5]).

The `boxes.mp` macros also provide for circular and oval boxes. These are a lot like rectangular boxes except for the shape of the bounding path. Such boxes

are set up by the `circleit` macro:

`circleit`⟨box name⟩`(`⟨picture expression⟩`)`

The `circleit` macro defines pair variable just as `boxit` does, except that there are no corner points ⟨box name⟩`.ne`, ⟨box name⟩`.sw`, etc. A call to

`circleit.a(...)`

gives relationships among points `a.c`, `a.s`, `a.e`, `a.n`, `a.w` and distances `a.dx` and `a.dy`. Together with `a.c` and `a.off`, these variables describe how the picture is centered in an oval as can be seen from the Figure 15.

The `drawboxed` and `bpath` macros work for `circleit` boxes just as they do for `boxit` boxes. By default, the boundary path for a `circleit` box is a circle large enough to surround the box contents with a small safety margin controlled by the internal variable `circmargin`. Figure 16 gives an example. The oval boundary paths around "Start" and "Stop" in the figure are due to equations of the form

⟨box name⟩`.dx` = ⟨box name⟩`.dy`

that force those boxes to be noncircular.

4.2 Other Packages

Why aren't the `boxes.mp` macros automatically preloaded like the plain macros? One reason is that they are too specialized to really be treated as part of the core language. Another reason is that `boxes.mp` was intended to be the first of several macro packages, each one extending the language to cover another specialized application.

In fact, there already is another macro package called `rboxes.mp`. This package builds on `boxes.mp` by providing another box shape: a rectangular box with rounded corners. Other box shapes could also be provided if there were a demand for them.

There should also be a macro package for drawing graphs. No such package has been designed yet, but there have been promising preliminary experiments with the automatic generation of axis labels for uniform and logarithmic spacing.

5 Conclusion

Building on METAFONT has made MetaPost a very powerful and flexible graphics language. It is especially well suited to generating figures in technical documents which may involve mathematical constraints that are best expressed symbolically. Such figures lack the aesthetic requirements that make font design so challenging.

This paper has introduced the MetaPost language via examples concentrating on interesting features that distinguish the language from other graphics languages and from METAFONT. Readers who want to use the language should refer to [3]. The MetaPost interpreter is currently available to academic institutions under non-disclosure agreement.

References

[1] Leslie Carr. Of METAFONT and PostScript. In *TEX User's Group Eighth Annual Meeting Conference Proceedings*. TEX User's Group, Providence, Rhode Island, 1988.

[2] John D. Hobby. A METAFONT-like system with PostScript output. *Tugboat, the TEX User's Group Newsletter*, 10(4):505–512, December 1989.

[3] J. D. Hobby. A user's manual for MetaPost. Computing Science Technical Report no. 162, AT&T Bell Laboratories, Murray Hill, New Jersey, April 1992. Can be obtained by mailing "`send 162 from research/cstr`" to `netlib@research.att.com`.

[4] Alan Jeffrey. Labelled diagrams in METAFONT. *TUGboat, Communications of the TEX User's Group*, 12(2):227–229, June 1991.

[5] Brian W. Kernighan. Pic—a graphics language for typesetting. In *Unix Research System Papers, Tenth Edition*, pages 53–77. AT&T Bell Laboratories, 1990.

[6] D. E. Knuth. *METAFONT the Program*. Addison Wesley, Reading, Massachusetts, 1986. Volume D of *Computers and Typesetting*.

[7] D. E. Knuth. *The TEXbook*. Addison Wesley, Reading, Massachusetts, 1986. Volume A of *Computers and Typesetting*.

[8] Richard O. Simpson. Nontraditional uses of METAFONT. In Malcom Clark, editor, *TEX Applications, Uses, Methods*, pages 259–271. Ellis Horwood, 1990.

[9] Michael J. Wichura. *The PiCTEX Manual*. TEX User's Group, Providence, Rhode Island, 1987.

[10] Shimon Yanai and Daniel M. Berry. Environment for translating METAFONT to PostScript. *TUGboat, Communications of the TEX User's Group*, 11(4):525–541, November 1990.

TEX for Everyone !?*

Theo Jurriens

Kapteyn Astronomical Institute
University of Groningen
P.O. Box 800
9700 AV Groningen
The Netherlands
taj@kapteyn.astro.rug.nl

Abstract

In this article author tries to defend a more general use of TEX outside the world of mathematics, astronomy, physics etc. The ! or ? in the title of this paper is the question.
Several examples are shown why TEX is much powerful than a dull word-processing package.

Keywords: METAFONT, PostScript, organisation, education

1 General

In this article TEX and LATEX have the same meaning. TEX or LATEX are extremely powerful tools to do your favourite job. Author prefers LATEX because for non-scientists (secretaries) it's much easier to learn then plain TEX.

The use of TEX is widely discussed in national and international magazines. An often forgotten fact of TEX is the stability of the program. There are not a bunch of updates every month. The change to TEX3 was the most relevant and important change in the last years.

The increase of e-mail will encourage the growth of TEX users. The latter is the best tool to prepare documents and exchange them machine-independent.

2 METAFONT

METAFONT is a difficult tool to make your own character-sets. In several journals, a lot of examples are given concerning the use of TEX in oriental languages. Impressive but once again coding both in the METAFONT source-file or the TEX input-file is not easy. Although I am not an expert in METAFONT, I was able to design a font for printing the phases of the Moon.

1992 sep 14
1992 sep 15
1992 sep 16
1992 sep 17
1992 sep 18
1992 sep 19
1992 sep 20

An example of including PostScript files within a TEX/LATEX document.

3 PostScript

In the last years the use of PostScript increased a lot. The use of PostScript fonts and including of graphics has given TEX new horizons. Since a few months I am editor of a local newsleaflet concerning the well-being of the inhabitants of my apartment building. The building is called oyster-catcher. Of course a proper picture, drawing of this bird may not be left-out.

Recently I used TEX and PostScript to design a poster for announcing a public lecture.

Advantages of using PostScript:

- An easy way to include graphics,
- In case of using PS fonts, small file sizes,
- A large collection of fonts, at unlimited sizes, is available/possible (see fig 1),
- High-quality output on film and paper is available,
- The bag of tricks (rotating text, shadowing, a better picture enviroment) is big.

*Presented at EuroTEX '92, September 14–18, Prague, Czechoslovakia.

```
\font\test=rptmr at 5pt
```

5pt

```
\font\test=rptmr at 25mm
```

25mm

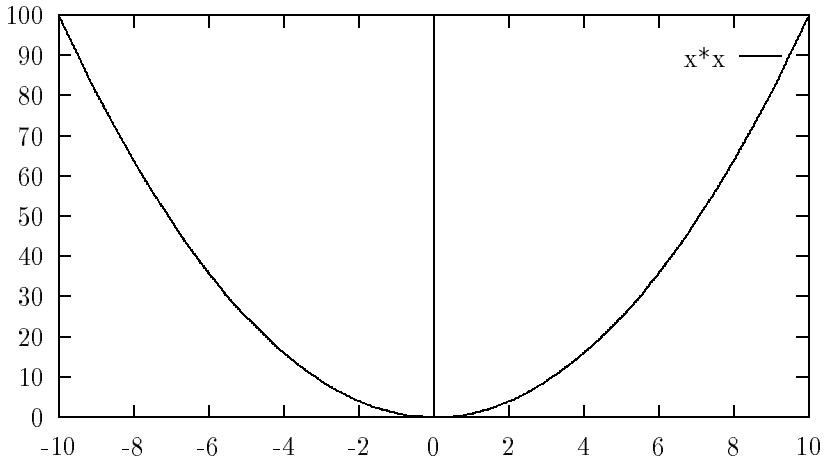
```
\font\test=rptmr at 0.75in
```

0.75in

Figuur 1: *Example of several fontsizes within PostScript.*

4 Graphics

Due to the increase of fast-machines/X11 based, doing and including graphics is simplified a lot. Also a lot of standard packages are able of producing PostScript-files. They can be included as described in the previous section. GNUPLOT is a tool available on several platforms. The graph, in figure 2 is made with this program (output: LaTeX picture environment file).



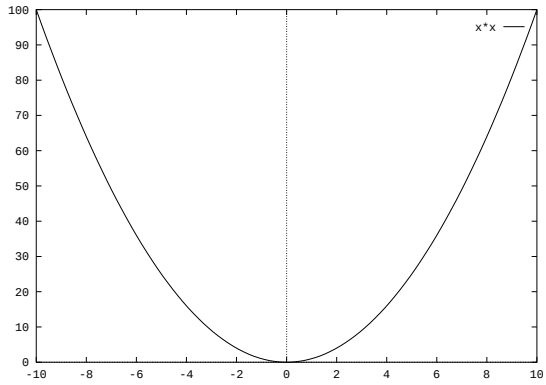
Figuur 2: *The graph $f(x) = x^2$ made with GNUPLOT*

A disadvantage is the rather big files produced by GNUPLOT. The file belonging to graph:

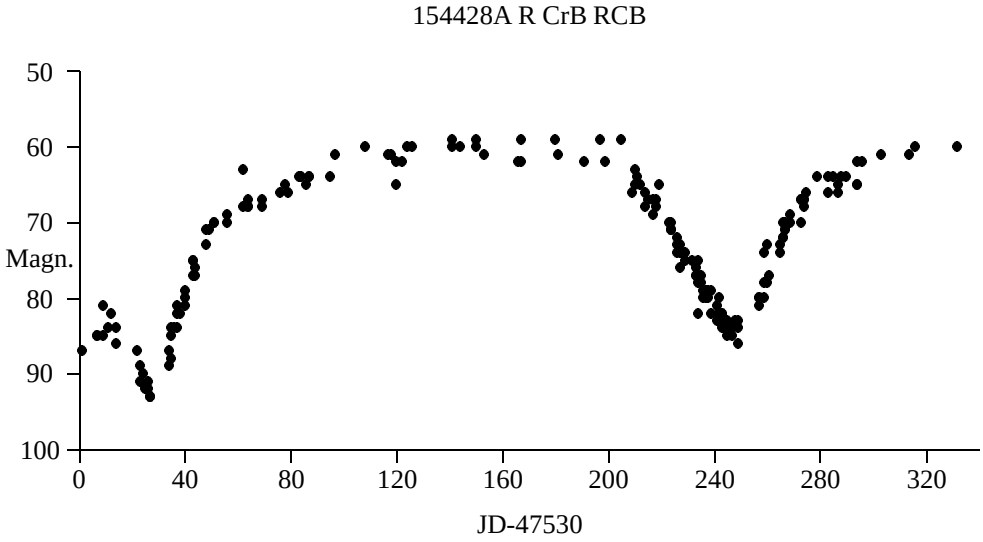
```
-rw-r--r--   1 taj   kapteyn 42555
             Jul 23 11:31 test.gnu
```

is big. GNUPLOT can also produce directly PostScript. In that case the files are much smaller.

```
-rw-r--r--   1 taj   kapteyn  5313
             Jul 23 11:39 gnuplot.eps
```



Figuur 3: *The same plot as Figure 2, now included as a postscript file.*



Figuur 4: *This plot is made with the help of PICTEX.*

4.1 PICTEX

Also PICTEX is a useful tool but requires a lot of memory and a high level of users experience. The plot in Figure 4 is made with PICTEX.

5 Education

It's often forgotten but education is needed to spread TEX around the world and around all different users. Concerning the training of secretaries I set up the 'Golden' rules (see previous MAPS).

References

[1] Theo Jurriens, "TEXniques in Siberia", in *Proceedings of the sixth EuroTEX Conference*, P. Louarn Ed., *Cahiers GUTenberg*, no. 10-11, September 91, pp. 7–13.

TEX als Database

Theo A. Jurriens

Kapteyn Astronomical Institute
P. O. Box 800
9700 AV Groningen
The Netherlands
taj@astro.rug.nl

Abstract

In dit artikel wordt beschreven hoe TEX is gebruikt als een primitieve database voor de administratie van de 47ste Nederlandse Astronomen Conferentie.

1 Inleiding

Een aantal gegevens zijn voor een conferentie-organisator van essentiëel belang:

- naam, adres etc.
- voordracht,
- tijd en zonodig plaats van voordracht,
- abstract,
- het te betalen bedrag en of men wel of niet betaald heeft.

Door deze gegevens van te voren goed te omschrijven kan een model-file gemaakt worden voor de verwerking van de gehele administratie. Iedere deelnemer 'kreeg' zijn eigen file met naam: `achternaam_initialen.nac`. Om verwarring te vermijden heb ik een niet `.tex` extentie gekozen. Voor de verwerking is een UNIX systeem gebruikt, zonder beperking in lengte filenaam.

2 Template

Het onderstaand template of model-file is gebruikt:

```
\naam{}
\staf{}
\kamergenoot{}
\adres{}
\postcode{}
\plaats{}
\instituut{}
\email{}
\voordracht{}
\poster{}
\wensen{}
\opmerkingen{}
\prijs{}
\betaald{n}
\tijd{n}{n}{n}
```

De instituutscode is ook gebruikt om een eenduidige adressering te realiseren. Bij het invoeren van een nieuwe deelnemer werd dus het template ingevuld. Met behulp van UNIX scripts en verschillende `.tex` macros werden verschillende lijsten gemaakt ten behoeve van de conferentie, zoals: deelnemerslijsten, lijst van voordrachten en badges.

Hieronder volgt het script voor het maken van een deelnemerslijst. De file `macros.tex` is aan het eind van dit artikel weergegeven. Het UNIX commando `cat *.nac` zorgt voor het appenden van de files aan de file met macro's in **alfabetische volgorde**.

```
cp macros.tex klad.tex
cat *.nac >> klad.tex
tex klad
dvihp -nodialog klad
spr klad
```

3 Macro's

Hieronder volgt de gebruikte set van macro's:

```
\parindent=0mm
\newcount\aantal
\newcount\totaal
\aantal=0
\def\naam#1{\gdef\Naam{#1}}
\def\kamergenoot#1{\gdef\Kamergenoot{#1}}
\def\staf#1{\gdef\Staf{#1}}
\def\adres#1{\gdef\Adres{#1}}
\def\postcode#1{\gdef\Postcode{#1}}
\def\plaats#1{\gdef\Plaats{#1}}
\def\instituut#1{\gdef\Instituut{#1}}
\def\email#1{\gdef\Email{#1}}
\def\voordracht#1{\gdef\Voordracht{#1}}
\def\poster#1{\gdef\Poster{#1}}
\def\wensen#1{\gdef\Wensen{#1}}
\def\prijs#1{\gdef\Prijs{#1}}
\def\opmerkingen#1{\gdef\Opmerkingen{#1}}
\def\betaald#1{\gdef\Betaald{#1}}
\def\tijd#1#2#3{\global\advance\aantal by 1
```

```
\vbox{
\Naam\ \Instituut
\if\empty\Staf \else\ \Staf \fi
\if\empty\Kamergenoot \else
\hfill \Kamergenoot \fi
\par
\Adres \hfill\break
\Postcode\ \Plaats \hfill\break
\if\empty\Email\else \Email\hfill\break\fi
\if\empty\Voordracht\else
Voordracht: \Voordracht\hfill\break\fi
\if\empty\Poster\else
Poster: \Poster\hfill\break\fi
\if\empty\Wensen\else Wensen:
\Wensen\hfill\break\fi
\if\empty\Opmerkingen\else
Opmerkingen: \Opmerkingen\hfill\break\fi
}}
```

Duidelijk zal zijn dat in combinatie met verschillende scripts en macro's er een flexibele werkomgeving ontstaat voor de registratie en mailing van conferentiedeelnemers.

LADIES and LATEX– III
Vragen allerlei!

Theo A. Jurriens

Kapteyn Astronomical Institute
P. O. Box 800
9700 AV Groningen
The Netherlands
taj@astro.rug.nl

Abstract

In het dagelijks gebruik van LATEX komen we zo af en toe toch nog wel eens problemen voor. In dit artikel vragen uit de praktijk en de mogelijke oplossingen.

1 Floats

Figuren of tabellen bij gebruik van `\begin{figure}` of `\begin{table}` drijven nog wel eens naar het eind van het artikel. Dit wordt veroorzaakt door het feit dat LATEX meent dat maar een bepaald deel van de pagina "float"mag zijn. De waarden kun je zelf veranderen door de volgende regels in de preamble te plaatsen:

```
\renewcommand{\textfraction}{0.05}
\renewcommand{\topfraction}{1}
\renewcommand{\bottomfraction}{1}
```

Uit de `latex.tex` file volgen de volgende definities:

- `\topfraction` : Fraction of column that can be devoted to floats.
- `\bottomfraction` : Same as above for bottom of page.
- `\textfraction` : Minimum fraction of column that must contain text.

Ook het plaatsen van de "floatëen alinea eerder of later kan veel helpen. Mijn eigen ervaring met de floats is dat ik de plaatsing van floats in eerste instantie aan LATEX over laat, de probleem gevallen plaats ik zelf met behulp van `[t]`. Immers tussen de "vleeshaken"[1] kunnen we de plaatsing van figuren aangeven.

Als het echt niet wil kunnen we ook twee figuren in één float plaatsen. Zie de figuren op bladzijde 103. Het is op de volgende wijze gemaakt, let op er is hier gebruik gemaakt van `figure*` in verband met een figuur welke twee kolommen overspant:

```
\begin{figure*}[t]
\vspace*{5cm}
\hbox{\parbox[t]{0.45\textwidth}{
\caption{De figuur links op deze pagina.
Toch mooi zo'n lege ruimte.
Treffen we niet meer aan in Nederland.}}
\hspace*{0.1\textwidth}
\parbox[t]{0.45\textwidth}{\caption{De
figuur rechts op deze pagina.}
\label{demo}}}
\end{figure*}
```

Bekend is dat je met `\caption{}` het bijschrift kunt plaatsen. Je bepaalt zelf of het boven of onder de figuur of tabel komt. Bij een bijschrift langer dan $\pm$ 300 tekens kan er een vervelende fout optreden: memory capicity exceeded. Oorzaak de `.lof` kan niet meer gelezen worden. Oplossing gebruik tussen de inmiddels bekende vleeshaken de verkorte tekst van het bijschrift voor gebruik in de "list of figures".

2 Twee-koloms

In veel voorkomende vraag is hoe krijg ik een kopregel over twee kolommen. Weer door middel van de vleeshaken:
`\twocolumn[`*tekst*`]`

Op deze wijze wordt de tekst over twee kolommen geplaatst, kan ook een alinea tekst zijn.

In een twee-koloms kunnen tabellen nog wel eens snel te breed worden. De oplossing is simpel: met `@{ }` het kolomwit vast zetten op een spatie of eventueel nog minder.

[1] Met dank aan Gineke Alberts: op ons laboratorium is een fraai jargon ontstaan: de `[]` noemen we vleeshaken, de {} noemen we flippers.

Figuur 1: *De figuur links op deze pagina. Toch mooi zo'n lege ruimte. Treffen we niet meer aan in Nederland.*

Figuur 2: *De figuur rechts op deze pagina.*

In twee- of meerkolommen tekst is het raadzaam om `\raggedright` te gebruiken, pas ook de grootte van de letter aan.

3 Serie-brieven

Het onderstaand voorbeeld toont hoe we serie-brieven maken.

```
\documentstyle[kapteyn]{letter}
\email{TAJ}
% je toestel nummer
\telefoon{4073}
\begin{document}
% nederlandse benaming in brieven
\dutch
\def\mailing#1{\begin{letter}
{#1}
%
%briefnummer
% briefnummer invullen
\nr{015}
% je ondertekening
\signature{Prof. Dr. P.C. van der Kruit\\
Voorzitter Afdeling Sterrenkunde}
% betreffende
\refer{uitnodiging}
%
\opening{\ }
Onze oud-hoogleraar Prof. Dr. A. Blaauw
heeft onlangs de
geschiedsschrijving van de E.S.O. voltooid.
Dit heeft geresulteerd in een door de
ESO uitgegeven boek
wat onlangs is verschenen.

Voor nadere vragen kunt U
terecht bij Theo Jurriens (050-634073).
% je tekst
\closing{Met vriendelijke groeten,}

\ps
P.S. Het Zernike complex is uitstekend
met het openbaar vervoer bereikbaar.
% bijlage
\encl{informatie-brochure ESO}
\end{letter}}
%
\mailing{Nieuwsblad
van het Noorden\\
Redaktie Wetenschap\\
Kees Wiese\\
Postbus 60\\
{\bf 9700 MC GRONINGEN}}
\mailing{Eddy Echternach\\
Ratelaarweg 31\\
9753 BE Haren}
\end{document}
```

Etiketten maken we op een andere wijze. Uiteraard is het gebruik van venster-enveloppen sterk aan te bevelen.

Onze `kapteyn.sty` kent ook een telefax optie. Het basislettertype is dan groter i.v.m. kans op onleesbaarheid door fax-transmissie.

Bij het maken van de brieven gebruiken we een Unix script. Bij het starten van een nieuwe brief wordt een model brief geladen: vrijwel indentiek als vorig voorbeeld.

4 Inhoudsopgave

In een boek moet, volgens de spelregels van de typografie, de inhoudsopgave romeins genummerd worden. Onderstaand voorbeeld toont hoe dit gaat (ontleend aan het jaarverslag van het Kapteyn Instituut):

```
\documentstyle[symbol,bk9pt,jaar,a5]{book}
\parindent=5mm
\widowpenalty=10000
\clubpenalty=5000
\renewcommand{\textfraction}{0.05}
\renewcommand{\topfraction}{1}
\renewcommand{\bottomfraction}{1}
\pagestyle{plain}
\begin{document}
% eerste pagina's romein
\setcounter{page}{1}
```

```
\renewcommand{\thepage}{\roman{page}}
\tableofcontents
\newpage
% opnieuw beginnen in arabisch
\setcounter{page}{1}
\renewcommand{\thepage}{\arabic{page}}
\include{foreword}
\include{bestuur}
\include{onderwijs}
\include{cluster}
\include{software}
\include{roden}
\appendix
\include{bijl1}
\include{bijl2}
\end{document}
```

Heeft de lezer ook leuke problemen (en misschien ook oplossingen) dan hoor ik het graag.

Just give me a Lollipop (it makes my heart go giddy-up)*

Victor Eijkhout

Department of Computer Science
University of Tennessee at Knoxville
Knoxville TN 37996-1301
eijkhout@cs.utk.edu

Abstract

The Lollipop format is a meta-format: it does not define user macros, but it contains the tools with which a style designer can easily implement such user macros. This article will show some of the capabilities of Lollipop and will give the reader a small peek behind the scenes of the implementation.

TEX is intended to support higher-level languages for composition

Donald Knuth

1 Introduction

One of the reasons that TEX is not widely accepted outside the scientific world is that the effort needed to create new visual designs, or even to make minimal modifications of a given design ("this article is a bit too long, but since we have rather generous margins, why don't we put the title in the margin next to the abstract, instead of over it") is disproportionally large. In Eijkhout and Lenstra (1991) it was argued that one way of solving this problem would be to implement powerful tools that a style designer could use to program macros without ever programming in TEX itself. In effect, the style designer "needs only say what she wishes done" (Perlis) and the meta-format creates the macros that do this. This article describes such a meta-format: Lollipop[1].

Now, for those who wondered at the title of this article, the first half refers to an epigram by Alan Perlis, to be found on page 365 of *The TEXbook*; the second half derives from a sixties ditty by Millie Small. All other etymologies are erroneous, and severely frowned upon.

2 The Structure of Lollipop

The Lollipop format tries to provide tools that make programming macros as hard as using them. I will not discuss the use of the resulting macros in detail, but will focus on implementational matters.

2.1 Working with Lollipop.

In order to process a document in Lollipop there has to be a 'style definition' for that document. This definition, a sequence of Lollipop macro calls, can be in the document itself, it can be `\input`, or it can be contained in a format. The latter option of loading a style definition in Lollipop and dumping the result as a new format is encouraged for two reasons. First of all, it indicates better the separation between the work of the style designer and that of the user. Secondly—especially on old computers (say of the order of a 286)—processing the style definition for a complicated document can easily take one or two minutes.

2.2 The basic Lollipop macros.

The Lollipop format is partly a macro collection – and some of the more interesting utilities will be discussed below – and partly a tool box for defining macros. The tools are four macros for defining

- headings (`\DefineHeading`): the main characteristics of a heading are that it has a title, and that it should stay attached to the following text;
- lists (`\DefineList`): a list is characterized by the fact that it has items;
- text blocks (`\DefineTextBlock`): a text block is basically just a group, however, it is so general that lists and headings are really special cases of text blocks; and
- page grids (`\DefinePageGrid`): a page grid is (a macro that installs) an output routine.

Each of these macros[2] can have a large number of options.

*Presented at TUG '92, July 27–30, Portland, Oregon,USA.

[1] The Lollipop format is available for anonymous ftp from `cs.utk.edu`.

[2] There is in fact a fifth macro `\DefineExternalItem`, closely related to `\DefineList`; it will be treated below.

2.3 An example of the use of Lollipop.

Although a large number of examples would be necessary to give a representative sample of the possibilities of the Lollipop tools, here is one example to give the reader the basic idea. The following macro defines a heading `\SubSection`.

```
\DefineHeading:SubSection counter:i
    whitebefore:18pt whiteafter:15pt
    Pointsize:14 Style:bold
    block:start SectionCounter literal:,
          SubSectionCounter literal:.
          fillupto:levelindent title
    external:Contents title external:stop
    Stop
```

(The terms '`block`', '`external`' et cetera are called 'options'.) This definition specifies that subsections have a counter that counts in lowercase roman numerals, that there should be a certain amount of white space above and below it, and that it should be formatted in 14 point bold as the section counter, a comma, the subsection counter, a full stop, filling these counters up to the `\levelindent` width (to be explained below), and following this by the title. Also it specifies that the title should go to the contents file.

This macro `\DefineHeading` must be a pretty complicated object, don't you think? Well, here is the full definition[3]:

```
\@GenericConstruct{Heading}
\def\@DefineHeading{
    \@DefineStopCommand{\relax}
    \csarg\edef{\@name}{\@GEN@OPEN
        \the\@main@options@list
        \@GEN@CLOSE}
    }
```

where the auxiliary macro `\csarg` is defined as

```
\def\csarg#1#2{\expandafter#1%
    \csname#2\endcsname}
```

2.4 Definition of the `\Define` macros.

Since the `\Define...` macros are so much alike – many options are common to all of them – I let all of them be defined automatically by the same macro `\@GenericConstruct`. This defines `\DefineHeading` as a macro that will process a list of options (this part contains the common work for all constructs), and then call `\@DefineHeading` to do the actual definition.

A call `\DefineHeading:Section` will expand first of all to a call

```
\def\@name{Section}
```

As can be seen in the example above, this macro is then used to define `\Section` with an `\edef`. This `\edef` unpacks the token list `\@main@option@list` that has been constructed during option processing. Also, the macros `\@GEN@OPEN` and `\@GEN@CLOSE` contain lots of conditionals that may or may not cause code to be included in the definition of `\Section` depending on values of parameters that were set during option processing. This is explained further below.

2.5 Options.

Clearly, a large responsibility rests on processing the options. For instance, in the example above the option '`counter`' has to allocate the appropriate counter, but also set the test `\has@countertrue`.

Options can be general, such as the '`counter`' option (here `\xp` is `\expandafter`):

```
\@GenericOption{counter}{\has@counteryes
    \NewCounter:\@name
    \xp\add@mark@item\xp{\@name Counter}
    \CounterRepresentation:\@name=#1
    }
```

or they can be specific, such as the option controlling white space between items in a list:

```
\@ListOption{whitebetween}{....}
```

Generic options are defined as follows:

```
\def\@GenericOption#1{
    \append@to@list
        {@GenericOptions}{\\#1;}
    \csarg\def{Option@#1}##1##2}
```

for instance, for '`counter`' a macro `\Option@counter` is defined. The definition

```
\@GenericConstruct{List}
```

causes the definition of `\@ListOption`:

```
\def\@GenericConstruct#1{
    \csarg\def{@#1Option}##1%
        {\csarg\def{#1@##1}####1####2}
```

so that the '`whitebetween`' option causes the definition of a macro `\@List@whitebetween`.

Now let's say we are defining a heading, and we find the option '`foo`'. We then check whether a macro `\Heading@foo` is defined. If so, we execute it; if not we check for the existence of a more general macro `\Option@foo`. This is executed if it exists, and if it doesn't, we check whether `\foo` is a defined control

[3] Several pieces of code in this article have been simplified. Others however, such as the following, have been left intact to convey to the reader the idea that Lollipop is a sophisticated format.

sequence. If it is, we include the command `\foo` in the `\@main@options@list`, so that it will later be part of the definition of the heading we are defining; if it is not, we generate an error message.

3 The Basic Tools

In this section I will give a short overview of the capabilities of the four basic macros. First the block structure macros used in all of them are explained briefly.

3.1 Block structure.

Text blocks and lists are obvious candidates for environment macros that do grouping, so that values of `\leftskip`, `\parindent`, and whatever more can stay local. As I've argued in (Eijkhout, 1990) such macros can also handle spacing above and below the environment. Thus, Lollipop has two macros

```
\def\@GEN@OPEN{\outer@start@commands
    \begingroup \inner@start@commands}
\def\@GEN@CLOSE{\inner@end@commands
    \endgroup \outer@end@commands}
```

that induce grouping, and that perform the various actions needed at the boundaries of an environment. This also includes such common actions as handling counters and titles, placing marks, and defining labels for symbolic referencing.

For instance, if the macro currently being defined (if this is `\Section`, the macro `\@name` has that value) has a counter, that should be incremented. Therefore the macro `\@GEN@OPEN` contains a line

```
\ifhas@counter
    \noexpand\StepCounter:\@name
\fi
```

Recall that these macros are called inside an `\edef`, so `\Section` macro contains the line

```
\StepCounter:Section
```

only if the macro has indeed a counter.

In general, a macro `\foo` opening the environment will contain the code generated by `\@gen@open`, while a corresponding command `\foostop` contains the `\@gen@close` code.

Headings

Maybe somewhat surprisingly, a heading can be considered as an environment, namely as one where the heading command contains both the opening and closing commands of the environment. Titles are treated below.

Text blocks

Text blocks are environments that can span several paragraphs. They have explicit open and close commands. Text blocks are, for instance, a way of having a chunk of text be indented and perhaps labeled. As an example, here is the specification of the examples in (Eijkhout, 1992): they are indented, and the word 'Example' is set in italic over them.

```
\DefineTextBlock:example
    breakbefore:500 breakafter:1
    PushListLevel
    noindent begingroup Style:italic
        literal:Example endgroup
    par nobreak Indent:no
    text
    Stop
```

The option '`text`' indicates where the text of the block fits in the specification. Any options appearing after this option will result in code in the macro that closes the environment. For instance, here is a possible way of defining left-aligning display equations:

```
\DefineTextBlock:DisplayEq
    whitebefore:abovedisplayskip
    whiteafter:belowdisplayskip
    whiteleft:levelindent
    literal:$ displaystyle text
    literal:$
    Stop
```

The closing macro will be defined as

```
\def\DisplayEqstop{ ...
    $
    \endgroup ... }
```

where the dollar corresponds to the one after the '`text`' option.

Lists

The main point of interest about lists is the formatting of the item labels. The two main choices are

```
item:left ... item:stop
```

for left-aligning, and

```
item:right ... item:stop
```

for right-aligning labels. The label can for instance contain an '`itemCounter`', or an '`itemSign`', or even both. The item sign and the representation of the item counter are dependent on the level, and can be set by the designer.

An interesting option is '`description`'. If this option is used, all text following `\item` to the end of line will be taken as the label text. The LaTeX style description list can be implemented as

```
item:left Style:bold
  description Spaces:2
  item:stop
```

which is used as

```
\item The label
 and the next line is again normal
```

Abbreviated closing

Both lists and text blocks have an explicit closing command. Since such phenomena are properly nested, the format can very well figure out what to close if I tell it to close the current block. Therefore, the macro `\>` closes whatever list or text block is opened last, and `\>]` closes all lists and text blocks that are currently open.

Page grids

Definition of output routines is much easier in Lollipop than in plain TEX, but still it is the hardest part of working with Lollipop. Hence I will not go into full detail.

The most important option for page grids is '`text`'. It indicates that a page will use part of `\box255`. If this option does not appear, we are defining an output routine that does not use `\box255`. For such output routines the option `\nextpagegrid` is crucial: it tells TEX what output routine to take when the current one has output a page.

For instance, if left and right hand pages have a different layout, we could implement them as separate output routines:

```
\DefinePageGrid:leftpage
    nextpagegrid:rightpage
    ...
\DefinePageGrid:rightpage
    nextpagegrid:leftpage
    ...
```

The '`text`' option usually appears inside

```
    band:start text band:end
```

and it can occur several times there. For instance

```
    band:start text
      white:20pt text
      band:stop
```

defines a two column layout with a gutter width of 20 point.

Some of the options for page grid (height and width for instance) have a global significance, but for others it is recorded whether they appear before or after the '`text`' options. Depending on this, they become part of the header or the footer of the page.

When the output routine is invoked, Lollipop assembles any header or footer, and computes the remaining space for text. If this is not equal to the size of `\box255`, `\vsize` is reset, and `\box255` is thrown back to the main vertical list. This mechanism is an easy way to get pages with the same size if the size of the header or footer can vary.

Definition of output routines is in fact so easy in Lollipop that for title pages of chapters it is easier to write a special page grid, than to mess around with a lot of macros. Thus the line

```
\Chapter The second chapter\par
```

may look to the user as calling a macro, whereas in fact it installs a new output routine for the chapter title page. The way the title is handled is explained below.

4 Titles and References

The perceptive reader may have noticed in the definition of `\DefineHeading` above that the macro defined is not declared with a parameter. How then are titles handled?

Well, since in Lollipop not only headings, but also lists, text blocks, and page grids can have titles (but need not; every once in a while a heading without a title can be convenient, and output routines with titles are surprisingly useful, as I indicated above), the option '`title`' controls whether a construct actually has a title by setting a switch `\ifhas@title` to true. Definition of the actual heading macro then executes a line

```
\ifhas@title \@Titelize{\@name}\fi
```

where `\@Titelize` is a macro that takes a macro, and redefines it with an argument.

This redefinition trick can even be performed twice: if the macro has a counter, this should be referenceable. For some reason I decided against the LATEX approach of using `\label` commands: any command that can be referenced in Lollipop[4] accepts an optional parameter with the label key. For instance

```
\Section[definition:section] Notations
    and Definitions\par
```

gives the key '`definition:section`' to a section.

5 Indentation Levels

If lists of various types are used in a nested fashion, each next level is indented with respect to the previous one by a certain amount. Specifying these amounts can be done quite flexibly in Lollipop, and it is also made easy for the designer to have other indented material line up with these implied left margins (Braahms, Eijkhout and Poppelier, 1989).

[4] Not explained in this article is that the way something is referenced is also easily determined by the user. This makes it possible for instance to refer to chapters by name instead of by number.

On each level, a control sequence `\levelindent` indicates the amount by which the next level will be indented. Thus, letting `\parindent` be set equal to `\levelindent` at the start of a text block, will give nicely aligning indentations no matter at what level the block appears.

The value of `\levelindent` is determined by looking at the level number (say that this is 3), and checking whether a macro `\levelindentiii` exists. If so, the value of this is taken, if not, some default fraction of the value of `\basicindent` is taken. The style designer can set this `\basicindent`, and adjust individual levels by

```
\LevelIndent:3=25pt
```

or similar calls.

Lists are indented to the next level automatically, but in order to provide this functionality for other objects there exists an explicit

```
\PushListLevel
```

command. There is even a `\PopListLevel` command that has various uses. For instance, it can be used to realise 'suspended lists': the effect of

```
\item Some text\par
{\PopListLevel
\noindent Some text.\par}
\item Again an item
```

is that the 'some text' aligns with the text outside the list, instead of with the items in the list.

Popping and pushing list levels is also essential for correct formatting of external files; see below.

6 Marks

TEX's marks are a means of communication between routines that supply certain information (values of counters, titles), and the output routine. Since there is no way for the output routine to tell the rest of the macros which ones should pass information through marks, in Lollipop *everyone* puts their information (that is, titles and counter values) in marks. The output routine then selects with a simple call, for instance

```
\LastPlaced:SectionTitle
```

the value of `\SectionTitle` in the `\botmark`.

Let's look at the implementation of this. There is a list `\mark@items` that has the names of everything that goes in a mark. For instance, defining a heading `\Section` causes calls

```
\add@mark@item{SectionTitle}
\add@mark@item{SectionCounter}
```

These allocate token lists

```
\mark@toks@SectionTitle
\mark@toks@SectionCounter
```

which are to contain the title and the counter value, and which get their value from a command such as

```
\refresh@mark@item
   {SectionTitle}{The title}
```

whenever `\Section` is called. Everytime a mark item is refreshed, a new mark is placed on the page which contains the values of *all* mark token lists. The output routine then simply picks from this structure whatever information it needs.

7 External Files

Formats such as LATEX usually supply facilities for a table of contents, and maybe lists of figures and tables, but what if an author needs in addition a list of notations, one of definitions, and one of authors referenced? Lollipop takes the drastic approach, and provides none of these.

But it makes it easy for you to define them yourself.

7.1 User interface.

An external file is characterized by in an internal name, and a file name extension:

```
\DefineExternalFile:Contents=toc
```

This command does some initialization such as a call to `\newwrite`, and it creates a switch so that the user can specify with

```
\WriteContents:no
```

that the file is not to be overwritten in this run. A global switch

```
\WriteExtern:no
```

prohibits all external writes.

Next, commands such as `\Section` have to specify that they want to write out information. This is done with the option '`external`'. Usually, all that is written out is the title

```
    external:contents title external:stop
```

but other information can be written out too.

The hard part about external files is specifying how their information is to be typeset. Telling that a file needs to be loaded is simple:

```
\LoadExternalFile:Contents
```

For every command that writes information to an external file, the style definition needs to contain a call

```
\DefineExternalItem:Section file:Contents
    item:left Style:roman SectionLabel
        item:stop
    title Spaces:2 Style:italic Page
    Stop
```

where '`SectionLabel`' is the counter that was written out automatically for `\Section`, and '`title`' is whatever information was specified with the '`external`' option.

In effect, this defines the layout of a list that has only one item. Now we see another use for pushing indentation levels: contents items for subsections may need to be indented, but since they are a separate list on the outer level, we need to push them explicitly to the correct indentation:

```
\DefineExternalItem:SubSection
    file:Contents
    PushIndentLevel Style:roman
    item:right SectionLabel literal:.
        SubSectionLabel Spaces:1 item:stop
    title Spaces:2 Style:italic Page
    Stop
```

Note that a composite label is made here out of the section and subsection numbers.

7.2 Implementation.

External files are handled in much the same way they are treated in LaTeX: all information is written to the main auxiliary file, and this is loaded at the end of the run, in order to update the other external files.

Writing out titles and such means that these are subject to the usual expansion of `\write`. The LaTeX approach of letting the user put `\protect` commands has proved over time to be too error-prone, so I've decided to inhibit all expansion in titles.

8 Extendability of Lollipop

For each macro package, the question comes up 'but what if I want something that it cannot do?' The option mechanism of Lollipop can cope with this quite easily. Any option that is undefined is interpreted as a control sequence. Thus the style designer can incorporate arbitrary macros.

For instance, the title pages of *TeX by Topic* (Eijkhout, 1992) have quite elaborate headings, for which I programmed a separate macro `\ChapterHead`, which uses the (automatically generated) macro `\ChapterTitle`.

```
\def\ChapterHead
   {\hbox{ ...
          \PointSize:24 \Style:roman
          \ChapterTitle
    ...}
```

The macro `\Chapter` then uses this `\ChapterHead`:

```
\DefinePageGrid:Chapter
    NextPageGrid:textpage HasTitle:yes
    ...
    ChapterHead
    ... Stop
```

The undefined option '`ChapterHead`' generates a call to the macro `\ChapterHead`.

9 Goodies

It goes without saying that Lollipop has a sophisticated font selection scheme, a verbatim mode, and other assorted niceties. However, since these facilities are rather pedestrian, if rather useful, I will not discuss them here.

10 Conclusion

Lollipop is a long, complicated format. An article about it can only give a taste of its philosophy. I hope this piece has given the reader an idea of how macros can be generated automatically, according to the wishes of a style designer. People wanting to use Lollipop can get the software and a user's guide; people wanting to understand it will have for a while only this article and the code to go on.

As yet there is no real experience with Lollipop. I have used it myself for two books, but I am the author. I find it very easy to use, but if something goes wrong the errors can be mystifying in the extreme.[5] Error messages are still a major concern. Recall that macros are automatically defined and redefined, by macros that are themselves never explicitly defined. Still, I hope that the dynamic approach will catch enough user mistakes already in the definition stage for this format to be of value to non-TeXnicians wishing to use TeX.

References

[1] Braams, Johannes, Victor Eijkhout, and Nico Poppelier, "The development of national LaTeX styles", *TUGboat*, **10**(3), pages 401–406, 1989.

[2] Eijkhout, Victor, *A paragraph skip scheme*, *TUGboat*, **11**(4), pages 616–619, 1990.

[3] Eijkhout, Victor, *TeX by Topic*, Addison-Wesley, 1992.

[4] Eijkhout, Victor and Andries Lenstra. "The document style designer as a separate entity", *TUGboat*, **12**(1), pages 31–34, 1991.

[5] Perlis, Alan, "Epigrams on Programming", *ACM Sigplan Notices*, **17** (9), pages 7–13, 1982.

[5] And the macros themselves can become pretty big. While debugging, I discovered that TeX will 'only' `\show` the first 1000 characters of a macro. . .

Index Preparation for TEX Related Documents*

David Salomon

11835 Sapota Dr.,
Lakeside, CA 92040, USA
dxs@ms.secs.csun.edu

June 1992

A beta release of the MakeIndex program has recently become available for the Macintosh computer, and I immediately started using it to prepare the indexes of two new books. MakeIndex is easy to use with LATEX but, since I like to work with plain TEX, I have developed all the necessary macros from scratch. They are presented here for the benefit of anyone who wants a professionally looking index.

A few words about the use of MakeIndex are in order. Either LATEX or TEX is used to create a raw index file (the IDX file) with entries such as \indexentry{abc}{24}, \indexentry{xyz}{108}. MakeIndex then reads the file, sorts the items, merges multiple occurences of the same item on the same page, and creates the final, IND, file with entries such as '\item abc, 11, 24, 101' '\item xyz 15, 76, 108'. The user has to define macro \item to typeset an index item in any desired way, and the entire index is then typeset by \input doc.ind. MakeIndex supports sub- and subsub items (indicated by a '!'), and has many other features (see appendix 1).

It is generally agreed that index preparation is a complex job, and that it should not be fully automated. Certain things are best done manually. My original intent was, therefore, to develop simple macros that are easy to read, understand and modify for specific needs, yet can do most of the work. My experience so far indicates that they typically do more than 90% of the work, so only a small part of the total effort of index preparation needs to be done manually.

The macros described here are divided into two groups, those that write index items on the IDX file, and those that read the final, IND file, and typeset the final index. Following the tradition of *The TEXbook* (p. 423), the circumflex is turned into an active character and is used to indicate index items in the source document. A typical example is ^{abc}. The macros have to take into account the following:

1. Some index items are 'silent', they should be written on the IDX file but should not appear in the document. The macros accordingly accept either 1 or 2 arguments of the forms ^{abc} and ^[xyz]{abc} (but not ^{abc}[xyz]). The string xyz is not typeset but becomes part of the index item on the IDX file.
2. Many documents about TEX use the notation |abc| for verbatim listings. The user should thus be allowed to write ^|abc| (but not ^|abc|{..}). In order that they not interfere with the sorting, the vertical bars should be removed from the IDX file, and be reinserted in the IND file.
3. The notation |\abc| presents another special case. The user should be allowed to write ^|\abc|, and both the vertical bars and the '\' should be removed before sorting. However, because of the special way macro \getpar below reads its argument, the removed items cannot simply be reinstated. The solution is to convert such an index item, in the IND file, to the form \bs abc\\. When the IND file is \input, macro \bs adds the '\' and typesets its argument verbatim.
4. In a document about TEX, certain index items are preceded by a '\'. A good example is the item \TeX. This item should be sorted without the '\', to appear among the Ts. The '\' should thus be removed from this item in the IDX file prior to sorting, and should be appended back to the same item on the IND file before typesetting it.
5. Certain characters, such as '$' and '%', have special meaning in TEX. Since an index item may contain such characters, their special meaning should be temporarily suppressed when writing an item on the IDX file.

Point 5 above is handled by the \sanitize macro which changes the catcodes of most special characters to 'other'. Since '^' is used to indicate an index item, the catcodes of ^, ^^K and ^^A are not changed. \sanitize should be expanded before the arguments of the index macros are scanned. The only macros with arguments are \inxA#1 & \inxB[#1]#2. Note below how \begingroup and \endgroup are used to localize the effect of \sanitize.

*Presented at the 9^{e} NTG meeting, June 4, 1992, Amsterdam.

As a result, things such as '`\insert`', '`#52`' and '`52%`' can easily be written on the index file, while '`J\^orgen`' cannot.

Table 1 shows examples of index items. For each item, the way it is specified in the source file is shown, as well as the resulting text in the document, and the way the item is finally typeset in the index.

Source	Typeset in document	Typeset in index
`^[abc]`	abc	abc
`^[xyz]{abc}`	abc	xyzabc
`^[xyz]{ abc}`	abc	xyz abc
`^|abc|`[1]	`abc`	`abc`
`^|\abc|`[1]	`\abc`	`\abc`
`^[|\abc |]{xyz}`	xyz	`\abc` xyz
`^{\abc}`	replacement text of `\abc`	same
`^[|\abc|!xyz]{}`[2]	nothing	`\abc`, xyz[3]

[1] Note that there is no need for braces in this case

[2] Main argument should be empty.

[3] This is a subitem.

Table 1

To handle the different cases above, a small utility program, `IdxInd`, has been written. It should be run on the IDX file before `MakeIndex` reads it, and on the IND file after it is created by `MakeIndex`. The entire process involves the following steps:

- The document is typeset by TEX and file `doc.idx` is prepared.
- `IdxInd` reads file `doc.idx` and creates file `tmp.idx`. This is called the first pass.
- `MakeIndex` is run on `tmp.idx`. It creates the sorted file `tmp.ind`.
- The first and last lines of the IND file are `\begin{index}` and `\end{index}`. They should be removed manually, since they are only used by LATEX.
- `IdxInd` is run on `tmp.ind` and creates file `doc.ind`. This is the second pass.
- File `doc.ind` is `\input` and the indexing macros (the second set below) used to typeset the final index. Any special index items will show up in this step and will require manual intervention. Also, bad page breaks in the index pages will have to be corrected by the user at this stage, either by changing the penalties used in the macros, or by inserting penalties at strategic points in file `doc.ind`.

Table 2 shows how different types of index items appear in the source file and in the four index files involved.

type	in source file	on index file `doc.idx`	`tmp.idx`	`tmp.ind`	`doc.ind`
1	`^[abc]{}`	`abc`	`abc`	`abc`	`abc`
2	`^|abc|`	`|abc|`	`abc*`	`\item abc*,..`	`\item |abc|`
3	`^|\abc|`	`|\abc|`	`abc#`	`\item abc#,..`	`\item \bs abc\\`
3 (with sub item)	`^[|\abc|!xyz]{}`	`|\abc|!xyz`	`abc#!xyz`	`\item abc#,..` `\subitem xyz,..`	`\item \bs abc\\,..` `\subitem xyz,..`
4	`{^\abc}`	`\abc`	`abc&`	`\item abc&..`	`\item \abc`
4 with) sub item)	`^[\abc!xyz]{}`	`\abc!xyz`	`abc&!xyz`	`\item abc&..` `\subitem xyz,..`	`\item \abc` `\subitem xyz,..`

Table 2

Note that one should not have, in the same document, items such as `^[\abc]{}` and `^|\abc|`. They would be changed by pass 1 to `abc&` and `abc#`, respectively, and would therefore be considered different by `MakeIndex`. Instead of `^[\abc]{}` the author should write `^[|\abc|]{}`. It is also permissible to write `^[|\abc||bold]{}` (with three vertical bars). In such a case, `IdxInd` will only remove the first two bars (and the backslash).

The last example in table 2 is of a main item `|\abc|` and a subitem `xyz`. The entire item must be silent (placed in square brackets), and the pair of braces that follows should be empty.

When `IdxInd` removes a pair of vertical bars (or vertical bars and a '\', or just a '\') it stores a special code following the modified item, so that the removed characters can later be restored. In table 2 the codes shown are an '*', a '#' or an '&'. In reality, the program uses ASCII codes 04, 03 and 02, respectively, for this purpose. Those are the codes of control characters, and so don't show up in print.

Here is the first set of macros, those that write the index entries on the IDX file. Note the use of `\string\indexentry`. Using `\noexpand\indexentry` also works, but writes a space following `\indexentry`.

```
\newwrite\inx
\immediate\openout\inx=\jobname.idx

\def\Caret{\ifmmode\def\next{^}\else\let\next=\indexT\fi\next}
\catcode‘\^=\active \let^=\Caret

\def\makeother#1{\catcode‘#1=12\relax}
\def\sanitize{\makeother\ \makeother\\\makeother\$\makeother\&%
\makeother\#\makeother\_\makeother\%\makeother\~\makeother\|}

\def\indexT{\futurelet\new\macX}
\def\macX{\ifx\new[\let\Next=\inxB\else\let\Next=\inxA\fi
\begingroup\sanitize\Next}
\def\inxA#1{#1\finidx{#1}}
\def\inxB[#1]#2{#2\finidx{#1#2}}
\def\finidx#1{\gdef\wridx{\write\inx{\string\indexentry{#1}{\folio}}}%
\endgroup\wridx}
```

The document, with index items, should follow, ending with `\closeout\inx`.

Special index items: Control sequences and items in angle brackets should be handled in a special way. The control sequence `\abc` should be indexed as `^[\abc]{}` (silent). An item such as `<xyz>` should be indexed as `^[<xyz>]{}` (silent) or `^[<xyz>]{}\lr{xyz}` where `\lr` is a macro producing the brackets.

Since in practice we don't want the brackets and backslashes to participate in the sorting, we should move them to the right in the IDX file (thus `^[abc\]{}` `^[xyz><]{}`), run `MakeIndex`, and move them back to the left in the IND file. For most documents, such cases are rare, and can be handled manually. For documents on TEX, a program should be written to do this. For the index of *The TEXbook*, a special code was written on the raw index file (pp. 423–424) to indicate special items. After sorting, a special program was run to interpret the codes and add brackets, backslashes, etc.

The index itself is prepared in double-column format. The macros for double columns appear in *The TEXbook*, pp. 416–417 and should be personalized. Typically, the values of `\hsize` & `\vsize` should be changed, and the `\shipout` in macro `\onepageout` may have to be modified.

The second set of macros appears below. These are the indexing macros, which are deliberately kept simple, to make them easy to read and modify. They should be sufficient for most needs. Note the following:

- An item such as `^[auxiliary spaces|see{ties}]{}` should appear only once in the document (since no page numbers will be listed for it anyway). The macros shown here will work even if `^{auxiliary spaces}` appears several times, but the `see` must appear in the first occurence.
- An item such as `^[auxiliary spaces|seealso{ties}]{}` should, similarly, appear just once in the document, at the last time `auxiliary spaces` is indexed.

```
\newif\ifpagebreak\pagebreaktrue
\def\item{\begingroup\obeylines\getpar%
                \global\pagebreaktrue}
{\obeylines
\gdef\getpar#1
{\par\ifpagebreak\bigbreak\else\nobreak\fi
\hangafter=1 \hangindent=15pt #1\par
\global\leftskip=0pt \endgroup}}
\def\goodfil{\hfil\penalty-9\hfilneg}
              %similar to \filbreak [353]
\def\see#1#2{{\it see }#1} % If there is
                  % more than one \see or
% \seealso per item, manual intervention
% is necessary.
\def\seealso#1#2{#2; \goodfil{\it see
                                  also }#1}
\def\bold#1{{\bf#1}}
\def\indexspace{\par \vskip 10pt plus 5pt %
                          minus 3pt\relax}
\def\subitem{\par\pagebreakfalse%
                     \leftskip=7.5pt\item}
\def\subsubitem{\par\pagebreakfalse%
                      \leftskip=15pt\item}
```

Assuming that the double-column macros (from *The TEXbook*, pp. 416–417) are available, the index can now be produced by:

```
\pageno=<whatever>
\line{\bf\hfil Index\hfil\hfil}\vskip.1in
A quotation can be placed here

\begindoublecolumns
\tolerance=6000 \parindent=0pt
\input mybook.ind
\enddoublecolumns
\bye
```

Exercise: An index is normally divided into 26 groups, each of items starting with the same letter. Modify

`\indexspace` to typeset the letter preceding each group.

Answer: If we can assume that all 26 letter groups are present, this is easy. The modified macro is:

```
\newcount\Letter\Letter='101
\def\indexspace{\par\vskip...%
                      \advance\Letter1%
           {\bf\char\Letter}\vskip...}
```

and an additional `\indexspace` command has to be placed manually in the IND file before the first group. If some groups may be missing, the simplest solution is to add a parameter to `\indexspace`:

```
\def\indexspace#1{\par\vskip...
              {\bf#1}\vskip...}
```

and supply the argument manually. A general solution may use the following idea: Macro `\indexspace` sets a flag (a `\newif` variable) to indicate that a letter should be typeset before the next item. Macro `\item` should test the flag. If the flag is true, `\item` should isolate the first character of its argument, typeset it (with appropriate vertical spacing) and clear the flag.

Appendix 1: MakeIndex

`MakeIndex` expects an IDX input file with entries of the form `\indexentry{entry}{page number}`. The following are the main options that it recognizes:

- A '!' can be used for sub items and subsub items. There are no subsubsub items. The IDX file entry `\indexentry{abc!opq!xyz}{23}` is converted by `MakeIndex` to the following three entries in the IND file:
  `\item abc`
  `\subitem opq`
  `\subsubitem xyz, 23`
  The user should define macros `\item`, `\subitem` and `\subsubitem` to typeset the index in any desired way.
- A vertical bar is used to indicate a command. The IDX file entry
  `\indexentry{abc|bold}{23}`
  is converted by `MakeIndex` to
  `\item abc \bold{23}` on the IND file. Again, the user should define a macro `\bold` to typeset the page number in the desired way. Other common examples are
  `\indexentry{abc|see{xyz}}{23}` and
  `\indexentry{abc|seealso{xyz}}{23}`.
- An '@' is used to specify a print entry that's different from the sort entry. The IDX file entry `\indexentry{eleven@xi}{23}` will result in '`\item xi, 23`' placed among the E's in the IND file.

Appendix 2: IdxInd

This small utility is written in Modula 2 for the Metrowerks PSE compiler on the Macintosh computer, and is freely available from the author. It has two separate parts, pass 1 and pass 2. In pass 1 it reads an IDX file, scans each record for special characters, replaces them with ASCII codes 02, 03 or 04, and writes the record on a new IDX file, to be processed by `MakeIndex`. After `MakeIndex` creates the IND file, pass 2 is run, to restore the special characters.

The program looks for a pair of vertical bars and for a backslash. It distinguishes three cases:

- An IDX file entry contains a pair of vertical bars. This corresponds to a type 2 record in table 2. The bars are removed, and an ASCII 02 in inserted at the back of the entry.
- The entry contains a pair of vertical bars, the first of which is immediately followed by a backslash. This corresponds to type 3 in table 2. The three characters are removed, and an ASCII 03 is inserted.
- A backslash but no vertical bars. This corresponds to type 4 in table 2. Again, the backslash is removed, and an ASCII 04 is inserted.

Pass 2 performs two tasks. The first is to look for the special ASCII characters and restore the vertical bars and/or backslash. The second task is to look for long, multi-line records, and convert them to a single line. An index item that appears on many pages in the document, may end up as a long record, with many page numbers, in the IND file. Such an entry is broken, by `MakeIndex`, into several lines. It may typically look like:

```
\item abc 3, 11, 18, ...
125, 153, 167,...
180, 242, 394, ...
```

Our simple macros expect each line to start with `\item`, `\subitem` or something similar. The three lines in the above example have thus to be united into a single record, and this is done in pass 2 by removing the carriage return characters at the end of the first two lines.

Table Diversions*

Kees van der Laan

Hunzeweg 57,
9893 PB Garnwerd, The Netherlands
cgl@rug.nl

September 1992

Abstract

Characteristics of existing table macro collections are given. A kaleidoscope of tables—as next best to a taxonomy—is presented. Newly introduced is the class of bordered tables. Variations in print—ruled, nonruled, framed, nonframed, dotted, centered, flushed—can be obtained via the invoke of parameter setting macros; no modification of user mark up. Simultaneous row and column spans, partial rules, and dotted lines are dealt with. The listing of the macro `\btable`, with auxiliaries, is included.

Keywords: (Bordered) Tables, framed and non-framed, ruled and non-ruled, simultaneous row and columns spans, partial rules, connected cells, plain TEX, SGML, education.

1 Introduction

The formatting of tables is considered complex, time consuming and expensive. Table making via TEX, or LATEX, needs too complex descriptions for simple tables, except when the table can be formatted via `\settabs`. For `\halign` use, there is no default template provided by plain. The $\langle X \rangle$TEX macros I have seen don't reflect the logical structure of the table. An apparently simple change of representation, for example from a ruled into a non-ruled, or from a framed into a non-framed table, requires in TEX, and LATEX, some non-negligible effort.

Therefore, there is a need for an easy to use macro on top of plain's powerful $\backslash\langle X \rangle$`halign`-s, which accounts for the structure of the table.

Characteristics of existing table macros are given in the Existing TEX Table Packages section. In the Kaleidoscope section an anthology of tables is presented. Peculiar are the deterministic tables, and the tables which update memory. The section ends with the bordered table model. In the Bordered Table section the macro and examples of use are given. In the Blocks section blocks and connected cells are dealt with.

No landscape vs. portrait issues are discussed, nor the general use of rotated fonts. Tables which extend the page will not be discussed either. See for the latter for example supertabular.sty, [7], or longtable.sty, [4]. I also refrained from the issues which come from mapping the 'main vertical list' onto pages, under the restriction not to split tables over page boundaries.

2 Existing TEX table packages

Cowan, [5], provided a nice package in order to facilitate the formatting of simple tables. A main feature of his macros is that no template and no explicit number of columns, have to be provided. The template is created dynamically from the data and the 'attributes.' This is powerful and Hi-TEX on the one hand, but restrictive and time consuming on the other. The automatism can't be easily superseded, as opposed to a default repetitive template, which I have adopted. The functionality as demonstrated by his examples can be easily obtained via the bordered table macro.

Khanh, [8], introduced table macros which mainly deal with complex header rows. Because I consider those complicated header rows the exception rather than the rule, and because the material is not in the public domain, I will not consider it further.

Spivak, [23], provided some powerful macros for formatting tables separately from the main document. His merging, at the dvi level, of the formatted table into the main document, is Hi-TEX. He also provided some interesting examples of simultaneous row and column spans. Good, but complex, and perhaps too advanced for daily use. It is in the public domain.

*Presented at EuroTEX '92, September 14–18, Prague, Czechoslovakia; however different from proceedings EuroTEX '92 in the encoding of FIFO.

Lamport, [19], mainly provided facilities for placing a table appropriately within the context.[1] He also developed `\cline`, for partial horizontal rules, which is a powerful and a handy feature. I don't consider his way of framing nice, nor his way of specifying the ruling.

Hendrickson, [8], has an approach similar to mine. No bordered table model, however, and less pronunciated 'attributes.'

There exist more table macros. Of those I had access to, the ones discussed above seemed relevant to me. Now and then visual mark up packages appear, like Type and Set, [2]. With bars and visual mark up the table is specified on the screen, and from that TEX code is generated. However useful this might be within the 'conversion' of wordwhatever into TEX area, I don't like the resulting TEX encoding. It is clumsy. If such simple tables generate such an encoding then we are lost with really something a little more complex. Besides, the purpose of this paper is to introduce some new ways of formatting *simple* tables, next to the general accepted formatting TEXniques. The TEXbook considers alignment the general issue. I like to approach the issue from the structure point of view, and to couple it to what is already available via plain.[2] This work is a continuation of my earlier work on the SGML-TEX relation, [13].

3 Kaleidoscope

Some examples of special tables are given in order to illustrate the diversity. With respect to programming, the deterministic tables and the tables which update the computer memory, deserve special attention. The latter category is induced by *computer* typography.

- Pascal triangle, a deterministic table, [14]

$$\begin{array}{ccccc} & & 1 & & \\ & 1 & & 1 & \\ 1 & & 2 & & 1 \\ 1 & 3 & & 3 & 1 \end{array}$$

via

```
$$\displaylines{1\cr
  1\quad1\cr
  1\quad2\quad1\cr
  1\quad3\quad3\quad1\cr}$$
```

- Tower of Hanoi, deterministic, [15]. The process of replacement of the disks will be printed by the invoke of `\hanoi`$\langle n \rangle$, n an integer. No user mark up is needed.
- Young tableaux, irregular shape, [9], [25]

1	2	3	4
5	6	7	
8			

7	8	9	10
11	13		
16			

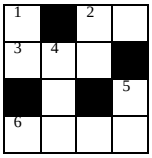
via

```
\def\young#1{\hbox{\vrule\vtop{\hrule\offinterlineskip\halign
{&\vbox{\hbox to\csize{\strut\hss##\hss\vrule}\hrule}\cr#1\crcr}}}}
%
\def\data{'&'&'\cr'&'&'\cr'}    \young\data\qquad
\def\data{7&8&9&10\cr11&13\cr16\cr}\young\data
```

with auxiliaries

```
\newdimen\csize\csize=3ex \newcount\cnt\catcode``=13
\def`{\global\advance\cnt1 \the\cnt}
```

- Much alignment occurs in typesetting mathematics: aligned equations in display, matrices, and the complex commutative diagrams. This has been dealt with elsewhere, for example [13], [14], and [23].
- Crosswords: puzzle, clues and solution, [16]



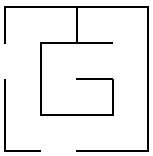
Across
2 Switch mode
3 Knuth
6 Prior to TEX

Down
1 Public domain
2 All right
4 All comes to it
5 Atari type

[1] As floating body, also called island by Spivak (1989).

[2] Jackie Damrau compiled a collection of table macros/styles resulting from the Portland workshop on the issue, which contains most of the works I mentioned above.

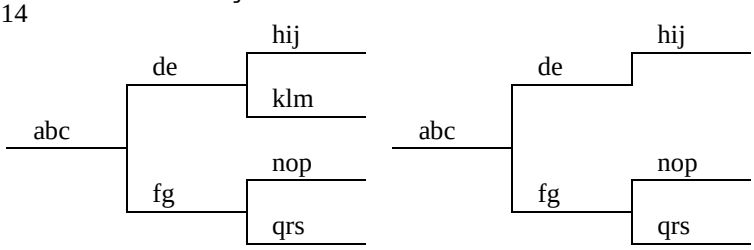
- Maze



via

```
\newdimen\csize \csize=3ex
\catcode`|=13 \def|{\vrule}
\catcode`_=13 \def_{\hrulefill}
\hbox{\vtop{\offinterlineskip\hrule
\halign{&\hbox to\csize
{\vrule height\csize width0pt#\hfil}\cr
          |&_&|_& \cr
           &|&_&  \cr
          |&|_&_&|\cr
          |_& &_&_\cr}%end \halign
 }\vrule%end \vbox
 }%end \hbox
```

- Chart, TEXbook ex22.14



via the following alternative code without `\halign`

```
\newdimen\dist \dist=2\baselineskip
\def\ent#1{\hbox{\vbox to0pt{\vss\hbox to10ex{\quad\strut#1\hss}\hrule}}}
\def\vl#1{\hbox{\vrule height#1\dist depth0pt}}
\def\bl#1{\kern#1\dist}%BLank of proper size
%Balanced
$$\offinterlineskip
 \vcenter{\ent{abc}}
 \vcenter{\ent{de} \vl2\ent{fg}}
 \vcenter{\ent{hij}\vl1\ent{klm}\bl1\ent{nop}\vl1\ent{qrs}}
\quad%Nearly balanced tree; pruning a branch
 \vcenter{\ent{abc}}
 \vcenter{\ent{de} \vl2\ent{fg}}
 \vcenter{\ent{hij}\vl{.5}%\bl{.5}\ent{klm}
                          \bl{1.5}\ent{nop}\vl1\ent{qrs}}$$
```

There exist some special tree packages, see [3], or consult Beebe's TUGlib server, especially the directory trees.bib.

- Row and/or column spans, partial horizontal rules[3]

Pair No	Contract		Results	Scores		MPs
	N–S	E–W		N–S	E–W	
1.						
2.						
et cetera						

- Bridge pair match scheme, with special 'first' element

Spel / Ronde	1–3	4–6	7–9	10–12	13–15	16–18	19–21
1.	1: 1–2			2: 3–4		3: 5–6	4: 7–8
2.	4: 8–3	1: 1–6			2: 5–7		3: 2–4
. . .							

[3] How many rows does the header have? Descriptive 1, visual 2, and for plain TEX formatting 3? See Mark up of Bridge Form subsection.

- Alignment at decimal points. Generally this is done by introducing an extra column for the dot, and flushing right the digits left and flushing left the digits right of it. It comes from the wishes not to print non-significant leading zeroes, and to suppress zeroes behind the point for exact numbers. 3.5 means exactly $3\frac{1}{2}$. 3.500 means accurate to three digits. I would mark up the numbers separately and insert Knuth's '?', TEXbook, p.240–241, for non-signicant zeroes, automatically. The latter can also be done by programming the editor by a template consisting of sufficient ?-s followed by the decimal point. The required numbers can be brought in by overtyping the template. Related to alignment at the decimal point is alignment at number signs,[4] see the AAP table.
- Fill-in forms, especially the registration forms for the various TEX conferences, are captivating. Sometimes I ponder about the e-mail equivalents of the traditional snail forms, especially what and how they should be filled in. Redefine a list of empty definitions?
- Time-tables (railway, bus, . . .), and the use of rotated fonts.
- Nested, and interrupted[5] table with updating of memory, [14]

Puzzle	♠ KQ76 ♡ J98 ♢ J942 ♣ 65	6NT, by East
♠ AJ3 ♡ K653 ♢ AK3 ♣ AQT	W N E S	♠ T9 ♡ A2 ♢ T5 ♣ KJ9xxxx
	♠ 8542 ♡ QT74 ♢ Q876 ♣ 2	

Trick					NS	EW
1	♡ 4!	♡ K	♡ 8	♡ 2	–	1
2	♣ A	♣ 5	♣ x	♣ 2	–	2
3	♣ Q	♣ 6	♣ x	♠ 2	–	3
4	♣ T	♡ 9	♣ K	♠ 4	–	4
5	♣ J	♠ 5	♠ 3	♠ 6	–	5
6	♣ 9	♠ 8	♡ 5	♠ 7	–	6
7	♣ x	♢ 6	♠ J	♢ 2	–	7

Puzzle	♠ KQ ♡ J ♢ J94 ♣ –	NS squeezed on ♣ continuation?
♠ A ♡ 63 ♢ AK3 ♣ –	W N E S	♠ T9 ♡ A ♢ T5 ♣ ⓧ
	♠ – ♡ QT7 ♢ Q87 ♣ –	

8	♣ x	♡ 7	♡ 6	♡ J	–	8

et cetera

Tables which also require updating of memory occur with typesetting of chess, for example [24], or GO, [11].

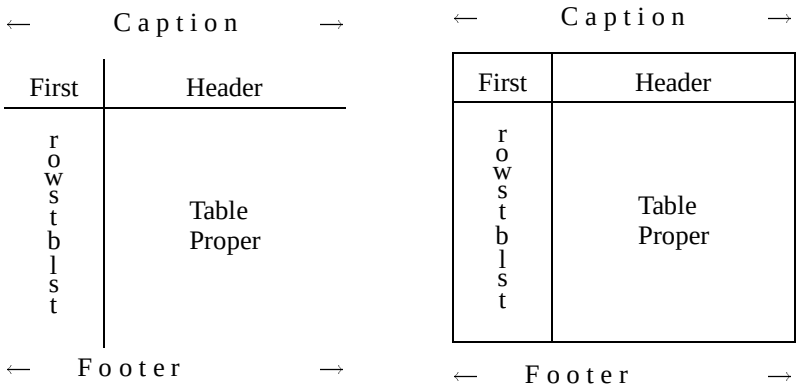
[4] Another column for the signs offends the structure. At a lower level one could think of `\scs`, a Sign Column Separator, but that is not nice either.

[5] From the user level one would say: aligned material connected by text, where the latter in the case at hand is the status of the play.

4 Bordered table

A bordered table consists of a caption, a header, row stubs, a table proper and a footer, all with appropriate separators. Special cases occur when some of the elements are absent. Abstraction from ruled and non-ruled is realized at the user level. Ruled can be specified either as horizontal, vertical or both. Dotted rules, also the partial ones, can be specified.[6] No variation in the line thicknesss has been encoded.[7] Framing is treated just as if a general document element has to be framed. I chose to leave the caption and the footer outside the frame. The positioning of the elements can be specified as centered, flushed left, respectively flushed right. A user may supply his own template. Caption, border (first, header, row stubs), and footer can be specified; all mutual orthogonal and independent from `\data`.

- Bordered table model



via

```
\def\caption{$\leftarrow$\hfill C a p t i o n\hfill$\rightarrow$}
\def\first{\hbox to6ex{\hss First\hss}}
\def\header{\hbox to18ex{\hss Header\hss}}
\def\rowstblst{{\hfil$\vcenter to20ex{\vss\offinterlineskip
              \fifo rowstblst\ofif\vss}$}}
\def\data{$\vcenter{\offinterlineskip\hbox{Table}\kern1ex\hbox{Proper}}$}
\def\footer{\hbox to\thsize
{$\leftarrow$\hss{\small F o o t e r}\hss\hss\hss$\rightarrow$}}
$$\vcenter{\btable\data}\qquad\qquad
  \vcenter{\framed\btable\data}$$
```

with, [17], [19]

```
\def\fifo#1{\ifx\ofif#1\ofif\fi\process{#1}\fifo}\def\ofif#1\fifo{\fi}
\def\process#1{\hbox to0pt{\hss#1\hss}\kern.5ex}
```

Rules can be specified via for example `\ruled`, `\hruled`, `\vruled`, `\nonruled`, respectively `\dotruled`.[8] The rules that separate the border can be eliminated, by redefinition of the macros `\headersep` and `\rowstbsep`. Furthermore, there are `\ctr`, `\fll`, and `\flr`, for controlling the positioning of the elements of the table. The framing can be controlled by `\framed`, `\dotframed`, or `\nonframed`.

4.1 Encoding

```
%btable.tex version 1, 17/7/92                          author: cgl@rug.nl
\newbox\tbl\let\ea=\expandafter
%Cell vertical size, row height and depth (separation implicit),
\newdimen\cvsize\newdimen\tsht\newdimen\tsdp\newdimen\tvsize\newdimen\thsize
%Parameter setting macros:   Rules
\def\hruled{\def\lineglue{\hrulefill}\def\colsep{}      \def\rowsep{\hrule}
   \let\rowstbsep=\colsep\let\headersep=\rowsep}
\def\vruled{\def\lineglue{\hfil}     \def\colsep{\vrule}\def\rowsep{}
   \let\rowstbsep=\colsep\let\headersep=\hrule}
\def\ruled {\def\lineglue{\hrulefill}\def\colsep{\vrule}\def\rowsep{\hrule}
   \let\rowstbsep=\colsep\let\headersep=\rowsep}
\def\nonruled{\def\lineglue{\hfil}   \def\colsep{}      \def\rowsep{}
   \def\rowstbsep{\vrule}\def\headersep{\hrule}}
\def\dotruled{\def\lineglue{\dotfill}\def\rowsep{\hbox to\thsize{\dotfill}}
```

[6] The chameleon sized `\hrule` and `\vrule`, don't have dotted analogues, however.

[7] It is not difficult to introduce a parameter `\linethickness`.

[8] The latter requires some extra work: table measurement via 2 passes.

```
\def\colsep{\lower1.5\tsdp\vbox to\cvsize{%
\leaders\hbox to0pt{\vrule height2pt depth2pt width0pt\hss.\hss}\vfil}}
\let\rowstbsep=\colsep\let\headersep=\rowsep}
%Parameter setting macros:   Controling positioning
\def\ctr{\def\lft{\hfil}\def\rgt{\hfil}}%Centered
\def\fll{\def\lft{}     \def\rgt{\hfil}}%Flushed left
\def\flr{\def\lft{\hfil}\def\rgt{}}     %Flushed right
%Parameter setting macros:   Framing
\def\framed{\let\frameit=\boxit}
\def\nonframed{\def\frameit##1{##1}}
\def\dotframed{\let\frameit=\dotboxit}
%
\def\btable#1{\vbox{\let\rsl=\rowstblst%Copy
\ifx\empty\template\ifx\empty\rowstblst
    \def\template{\colsepsurround\lft####\rgt&&\lft####\rgt\cr}
    \else\def\template{\colsepsurround####\hfil&&\lft####\rgt\cr}\fi
   \fi
\tsht=.775\cvsize\tsdp=.225\cvsize
\def\tstrut{\vrule height\tsht depth\tsdp width0pt}
%Logical mark up of column and row separators, via use of
\def\cs{&\colsepsurround\colsep\colsepsurround&}
\def\prs{&\colsepsurround\lineglue&}   \def\srp{&\lineglue\colsepsurround&}
\def\rs{\colsepsurround\tstrut\cr
        \ifx\empty\rowsep\else\noalign{\rowsep}\fi
        \ifx\empty\rowstblst\else\ea\nxtrs\fi}
\def\grs{\colsepsurround\tstrut\cr\ghostrow}
\def\rss{&\colsepsurround\rowstbsep\colsepsurround&}
\def\hs{\colsepsurround\tstrut\cr
       \ifx\empty\headersep\else\noalign{\headersep}\fi
       \ifx\empty\rowstblst\else\ea\nxtrs\fi}
\preinsert                                              %User action
\setbox\tbl=\vbox{\tabskip=0pt\relax\offinterlineskip
\halign{\span\template\ifx\empty\first\ifx\empty\rowstblst\else
\ifx\empty\header\else\ea\rss\fi\fi\else\first\ea\rss\fi
\ifx\empty\header\ifx\empty\first\if\empty\rsl\else\ea\nxtrs\fi
                \else\ea\hs\fi
\else\header\ea\hs\fi
#1\colsepsurround\tstrut\crcr}    }                              %end \setbox
\postinsert%Pick up what is needed, \thsize,...
\ifx\caption\empty\else\hbox to\thsize{\strut\hfil\caption\hss}\captionsep\fi
\frameit{\copy\tbl}
\ifx\footer\empty\else\footersep\hbox{\vtop{\noindent\hsize=\thsize%
\footer}}\fi                     }}                             %end \btable
%Defaults
\cvsize=4ex\tsht=.775\cvsize\tsdp=.225\cvsize\def\colsepsurround{\kern.5em}
\def\caption{}\def\first{}\def\header{}\def\rowstblst{}\def\footer{}\def\data{}
\def\captionsep{\medskip}    \def\headersep{\hrule}
\def\footersep{\smallskip}   \def\rowstbsep{\vrule}
\def\preinsert{}
\def\postinsert{\global\thsize=\wd\tbl
               \global\tvsize=\ht\tbl\global\advance\tvsize by\dp\tbl}
\ctr\nonruled\nonframed\def\template{}\def\ghostrow{}            %end Defaults
%Auxiliaries
\def\boxit#1{\vbox{\hrule\hbox{\vrule\vbox{#1}\vrule}\hrule}}
\def\dotboxit#1{\vbox{\offinterlineskip\hbox to\thsize{\dotfill}%
\hbox{\lower\tsdp\vbox to\tvsize{%
\leaders\hbox to0pt{\hss\vrule height2pt depth2pt width0pt.\hss}\vfil}%
\vbox{#1}\lower\tsdp\vbox to\tvsize{%
\leaders\hbox to0pt{\hss\vrule height2pt depth2pt width0pt.\hss}\vfil}}%
\hbox to\thsize{\dotfill}}}
%And to account for logical columns with \logmsp
\def\spicspan{\span\omit}
\def\logmsp#1{\omit\mscount=#1\multiply\mscount by2 \advance\mscount by-1
\loop\ifnum\mscount>1 \spicspan\advance\mscount by-1 \repeat}
```

```
%To handle the row stub list: \rsl
\def\nxtrs{\ifx\empty\rsl\else\def\nxtel{\ea\nrs\rsl\srn}\ea\nxtel\fi}
\def\nrs#1#2\srn{\gdef\rsl{#2}#1\rss}                          %end btable.tex
```

`\btable` Implements the bordered table model. Note that the table is measured in order to set the width of the header and the footer. The measurement is also useful for dotted horizontal lines. When no template is provided, then a default will be created, automatically.

`\`$\langle X \rangle$`sep` Parameters which govern the kind of separators.

`\nxtrs` Yields the next element of the row stub list. *The last element of the list has to be enclosed by an extra pair of braces.* I could have introduced an explicit list terminator instead, but that would have created incompatibilities with the termination in other defs. Weird?

`\cvsize, \tsht, \tsdp, \thsize, \tvsize` These dimension variables stand for vertical cell size, table strut height, table strut depth, table horizontal size, and table vertical size.

`\preinsert, \postinsert` For flexibility. Default `\preinsert` is empty. `\postinsert` delivers by default the table sizes globally into `\thsize`, and `\tvsize`.

4.2 Examples of use

Before the invocation of `\btable\data` do:
Redefine `\data` with column and row separators, `\cs`, respectively `\rs`.
Choose from the 'attributes'
– `\ruled`, `\hruled`, `\vruled`, `\dotruled` Default: `\nonruled`
– `\framed`, `\dotframed` Default:`\nonframed`
– `\fll`, `\flr` Default: `\ctr`
Supply the caption, header, first element, row stubs, or footer via redefinitions of the corresponding macros.

- Just data, [14]

2	7	6
9	5	1
4	3	8

16	3	2	13
5	10	11	8
9	6	7	12
4	**15**	**14**	1

via

```
$$\def\data{2\cs7\cs6\rs 9\cs5\cs1\rs  4\cs3\cs8}
  \vcenter{\framed\ruled\btable\data}\qquad\qquad\qquad
  \def\data{16\cs    3\cs    2\cs13\rs
            5\cs   10\cs   11\cs 8\rs
            9\cs    6\cs    7\cs12\rs
            4\cs\bf15\cs\bf14\cs 1}
\ruled\framed\setbox0=\btable\data%for measuring sizes
\vcenter{\dotruled\dotframed\btable\data}\qquad$$
```

- Data plus header, TEXbook p.246

Year	World Population
8000 B.C.	5,000,000
50 A.D.	200,000,000

Year	World Population
8000 B.C.	5,000,000
50 A.D.	200,000,000

via the simpler nearly descriptive mark up

```
\def\header{\hfill Year\hfill\cs\hfill World Population\hfill}
\def\data{8000 B.C.\cs  5,000,000\rs 50 A.D.\cs200,000,000}
$$\flr\vruled\vcenter{\btable\data}\qquad
\ruled\framed\vcenter{\btable\data}$$
```

- Data with row stubs, TEXbook p.232

Horizontal lists	Chapter 14	
Vertical lists	Chapter 15	via
Math lists	Chapter 17	

```
\def\rowstblst{{Horizontal lists}
  {Vertical lists}{{Math lists}}}
\def\rowstbsep{}%Default a \vrule
\def\data{Chapter 14\rs
Chapter 15\rs Chapter 17}
$$\btable\data$$
```

- Data, row stubs, header, caption and footer

11	12
21	22

	Header	
1^{st} row	11	12
2^{nd} row	21	22

Caption

	Header	
1^{st} row	11	12
2^{nd} row	21	22

Footer

Caption

	Header	
1^{st} row	11	12
2^{nd} row	21	22

Footer

after data definition, `\def\data{11\cs12\rs21\cs22}`, via[9]

```
$$\vcenter{\framed\btable\data}      \qquad
  \def\header{\logmsp2\hfill Header\hfill}
  \def\rowstblst{{$1^{st}$ row}{{$2^{nd}$ row}}}
  \vcenter{\btable\data}             \qquad
  \def\caption{Caption}\def\footer{Footer}
  \vcenter{\dotruled\btable\data}    \qquad
  \vcenter{\ruled\framed\btable\data}$$
```

- Walter's spreadsheet, TEXbook, p.244. No preamble has to be defined. It makes ex22.10 superfluous. How about that?

1	Adjusted gross income	$4,000
2	Zero bracket amount for	
	a single individual	$2,300
3	Earned income	1,500
4	Substract line 3 from line 2	800
5	Add lines 1 and 4. Enter here	
	and on Form 1040, line 35	$4,800

The above is encoded, as a 4-column table, via

```
\def\data{1\cs\logmsp2Adjusted gross income     \dotfill\cs\$4,000\rs
          2\cs Zero bracket amount for                          \rs
           \cs a single individual \dotfill\cs\hfill\$2,300\cs  \rs
          3\cs Earned income       \dotfill\cs
                                    \hfill\underbar{ 1,500}\cs  \rs
          4\cs\logmsp2Substract line 3 from line 2
                                  \dotfill\cs\hfill\underbar{ 800}\rs
          5\cs Add lines 1 and 4. Enter here                    \rs
           \cs\logmsp2and on Form 1040, line 35 \dotfill\cs\$4,800}
\def\colsepsurround{\kern.5ex}\cvsize=3ex
$$\fll\btable\data$$
```

[9] Note the invariance of the `\data` specification.

- Part of AAP's table, [1], or [13]

AAP's table: Job Changes 1973–1980

	Gain/Loss of hospitals since 1973	Total No of CEO Job Changes 1973–90	Survival Rate of CEO's
Texas	+20	—	22%
Maryland	+ 5	42	24%

Source: David Kinzer, 'Turnover Of Hospital Chief Executive Officers: A Hospital Association Perspective,' Hospital and health Services Administration May-June 1982.

obtained via

```
\def\caption{AAP's table:\qquad Job Changes 1973--1980\hfill}
\def\rowstblst{{Texas}{Maryland}}}
\def\header{\copy1\cs\copy2\cs\copy3}
\def\data{$+20$\cs---\cs22\%\rs
          $+?5$\cs 42\cs24\%}
\def\footer{Source: David Kinzer, 'Turnover Of Hospital Chief
Executive Officers: A Hospital Association Perspective,'
Hospital and health Services Administration May-June 1982.}
$$\btable\data$$
```

with auxiliaries

```
\def\nl{\hfil\strut\break\strut}\catcode'?=\active \def?{\kern1.1ex}
\setbox1=\vtop{\noindent\hsize14ex
    Gain/Loss of\nl hospitals\nl since 1973}
\setbox2=\vtop{\noindent\hsize13ex
    Total No\nl of CEO Job\nl Changes\nl 1973--90}
\setbox3=\vtop{\noindent\hsize10ex
    Survival\nl Rate of\nl CEO's}
```

- AT&T table diversions, TEXbook p.247, [20]

```
\def\caption{AT\&T Common Stock}
\def\header{Year\cs Price\cs Dividend}
\catcode'?=\active \def?{\kern1.1ex}
\def\data{1971\cs41--54\cs\llap{\$}2.60\rs
             2\cs41--54\cs         2.70\rs
             3\cs46--55\cs         2.87\rs
             4\cs40--53\cs         3.24\rs
             5\cs45--52\cs         3.40\rs
             6\cs51--59\cs         ?.95\rlap*}
\def\footer{* (first quarter only)}
$$\framed\vcenter{\vbox{\small\btable\data}}\qquad\qquad
\def\caption{}\def\header{\logmsp3 \hfil AT\&T Common Stock\hfil\rs
           \hfill Year\hfill\cs\hfill Price\hfill\cs \hfill Dividend\hfill}
\vcenter{\flr\ruled\btable\data}$$
```

yields

AT&T Common Stock

Year	Price	Dividend
1971	41–54	$2.60
2	41–54	2.70
3	46–55	2.87
4	40–53	3.24
5	45–52	3.40
6	51–59	.95*

* (first quarter only)

AT&T Common Stock		
Year	Price	Dividend
1971	41–54	$2.60
2	41–54	2.70
3	46–55	2.87
4	40–53	3.24
5	45–52	3.40
6	51–59	.95*

* (first quarter only)

It remains a matter of choice what should be considered as header and what as caption. A footnote facility, which will append the footnote to a footnote list similar to the row stub list, could have been implemented. I refrained from implementing it, for the moment.

- Pittman's, [22], deterministic multiplication table. Typographically and in other programming languages a trifle. In TEX the encoding matters.

1	2	3
2	4	6

×	1	2	3
1	1	2	3
2	2	4	6

×	1	2	3	4
1	1	2	3	4
2	2	4	6	8
3	3	6	9	12

via

```
\let\nx=\noexpand \let\ea=\expandafter
\newcount\rcnt \newcount\ccnt \newcount\tnum
\newcount\mrow \newcount\mcol
\rcnt1 \ccnt1 \mrow2 \mcol3
\def\rows{\global\ccnt1 \cols \global\advance\rcnt1
   \ifnum\rcnt>\mrow\swor\fi\rs\rows}
\def\swor#1\rows{\fi}
\def\cols{\tnum\rcnt \multiply\tnum\ccnt \the\tnum
   \global\advance\ccnt1 \ifnum\ccnt>\mcol\sloc\fi\cs\cols}
\def\sloc#1\cols{\fi}
\vcenter{\btable\rows}\qquad\qquad
%
\global\ccnt=1 \global\rcnt=1 \mrow2 \mcol3
\def\first{$\times$}  \def\header{\row}
\def\row{\the\ccnt\global\advance\ccnt1
   \ifnum\ccnt>\mcol\wor\fi\cs\row}
\def\wor#1\row{\fi}
\def\rowstblst{\ifnum\rcnt=\mrow\gdef\rowstblst{}\fi}
\def\nxtrs{\the\rcnt\rss}
\vcenter{\btable\rows}\qquad\qquad
%
\global\ccnt=1 \global\rcnt=1 \mrow3 \mcol4
\def\rowstblst{\ifnum\rcnt=\mrow\gdef\rowstblst{}\fi}
\framed\vcenter{\btable\rows}$$
```

4.3 No mark up

For simple tables the data can be specified without `\cs`-s, and `\rs`-s, that is without explicit markers for column and row separators. These can be inserted by TEX, see TEXbook p.249, [17], or [22]. For special cases, for example crosswords, [16], this is handy.

Pondering about lists, make you realize that the table proper data forms a nested list with `\rs` and `\cs`, respectively first level and second level separators. One could also adopt $\langle cr \rangle$ and ␣ for that, in other words allow natural data provision.

5 Simultaneous row and column spans

When one kind of spans is the issue one can use `\logmsp`—for logical columns, as an extension to `\multispan`—either within `\halign`, for column spans, or within `\valign`, for row spans. Simultaneous row and column spans require more work.
First, we have to account for the blocks proper. This is done via leaving space open, overprinting the left open space, and the use of a *Ghost Row Separator*. For a block of even *row* size, I chose to insert lines of height zero. I call these ghost lines. For a block with an odd number of rows, I chose to modify the middle line.
Next, when we are dealing with *ruled* tables we have to hide the rules that should otherwise traverse the block. This can be done for partial vertical rules via `\logmsp{n}\hfil`, with n an integer. For partial horizontal rules there are two aspects. First, the table-wide rules are not typeset, and second, partial horizontal rules have to be typeset around the blocks. I supplied the macros `\prs`, Partial Rule Separator, and `\srp`, *mirror* Separator Rule Partial, to enclose a partial rule. The kind of partial rule is determined by the replacement text of `\lineglue`.

Row spans in the row stub list can be handled similarly. For even spans define an appropriate `\ghostrow`, with a copy of the block in it. For odd spans provide a copy of the block in the row stub list. Provide suitable empty row stub elements, { }, in both cases.

5.1 Partial rules

For a horizontal rule over one (logical) column use
`...\prs\lineglue\srp...` instead of
`...\cs...\cs...`.
`\lineglue` is defined by the $\langle X \rangle$`ruled` macros.

The `\prs`, Partial Rule Separator, accounts for that part of the rule, which extends into the hidden column for the separator. `\srp` is its mirror terminator. At the line-ends of the table one can simply use `\cr`, as mirror terminator. The respective (symmetric) definitions are

```
\def\prs{&\colsepsurround\lineglue&}   \def\srp{&\lineglue\colsepsurround&}
```

For a horizontal rule over $\langle n \rangle$ (logical) columns use

```
...\prs\logmsp{n}\lineglue\srp...%n logical columns
```

5.2 Mark up bridge form

The header row of the bridge form can be characterized by: one 3-row span, two 2-column spans, and two 2-column partial horizontal rules. Furthermore, the repetition of the number of rows is interesting.

```
\def\header{\cs\logmsp2\hfil Contract\hfil\cs
                        \cs\logmsp2\hfil Scores\hfil\cs\grs
            \cs N--S\cs E--W\cs\cs N--S\cs E--W\cs}
\def\data{\lines}   \framed\ruled\btable\data
```

with auxiliaries

```
\newcount\bcnt
\def\lines{\global\advance\bcnt by 1 \ifnum\bcnt=3 et cetera\hidewidth
   \senil\fi\the\bcnt.\cs\cs\cs\cs\cs\cs\rs\lines}
\def\senil#1\lines{\fi}
%
\def\ghostrow{\omit\colsepsurround%
  \vbox to 0pt{\vss\hbox to 5ex{\hss Pair\hss}\vskip.5ex
          \hbox to 5ex{\hss No\hss}\vss}\prs\logmsp2\lineglue\srp
  \vbox to 0pt{\vss\hbox to 5ex{\hss Re-\hss}\vskip.5ex
        \hbox to5ex{\hss sults\hss}\vss}\prs\logmsp2\lineglue\srp
  \vtop to 0pt{\vss\hbox to 5ex{\hss MP$\,$s\hss}\vss}\colsepsurround\cr}
```

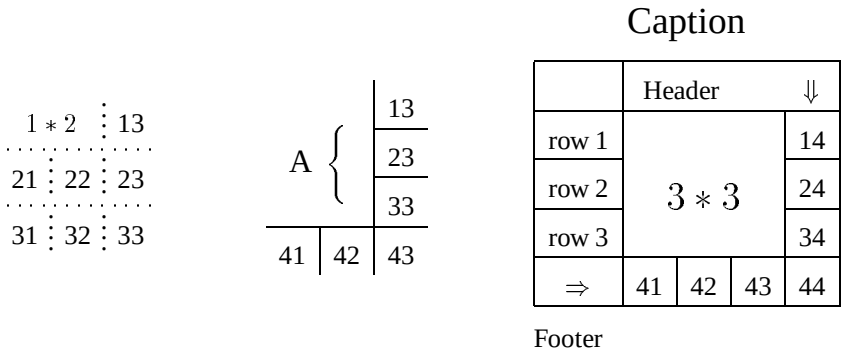
The row span extends implicitly into the first and third row. The last row of the data does not take vertical rules, automatically.[10]

[10] For a real form—24 or more rows—just modify `\lines`.

5.3 Blocks; connected cells

In order to specify spans of *logical* columns, we need an adapted version of `\multispan`, TEXbook p.354, as given in the Encoding subsection.
Spans form a block. Around the blocks partial rules emerge.

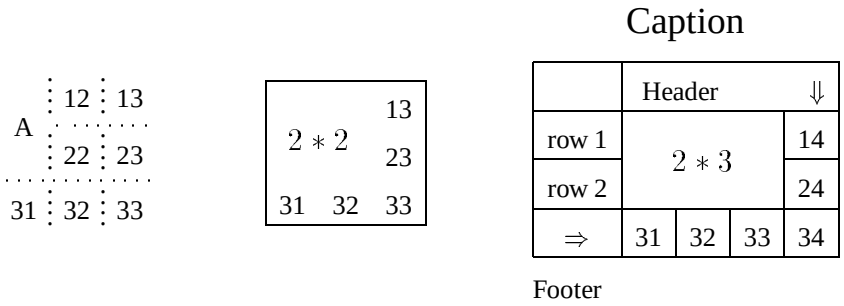
- **Odd blocks**, 1-by-2 etc. Use `\logmsp`, `\srp`, `\lineglue` and the like. Define appropriate -s.



can be obtained via

```
$$\setbox\block=\hbox{$1*2$}
\def\data{\logmsp2\colsepsurround\hfil\copy\block\hfil\cs13\rs
21\cs22\cs23\rs
31\cs32\cs33}                                              %end \data
{\ruled\setbox0=\btable\data}%For measurement
\vcenter{\dotruled\btable\data}\qquad\qquad
%
%3-by-2, and connected cells
\def\data{\logmsp2\hfil\cs13\grs
\logmsp2\colsepsurround\hfil\vbox to0pt{\vss
                                \copy\block\vss}\hfil\cs23\grs
          \logmsp2\hfil\cs33\rs
                   41\cs42\cs43}                           %end \data
\setbox\block=\hbox{{\large A}\enspace%
              $\left\{\vrule height3ex depth3ex width0pt\right.$}
\def\ghostrow{\ifx\empty\rowstblst\else\omit\lineglue\ea\srp\fi
\logmsp2\hfil\prs\lineglue\cr
\ifx\empty\rowstblst\else\ea\nxtrs\fi}
\vcenter{\ruled\btable\data}\qquad\qquad
%
%3-by-3 block and all around
\def\data{\logmsp3\hfil\cs14\grs
\logmsp3\hfil\vbox to 0pt{\vss\copy\block\vss}\hfil\cs24\grs
          \logmsp3\hfil\cs34\rs
              41\cs42\cs43\cs44}                           %end \data
\setbox\block=\hbox{\Large$3*3$}
\def\ghostrow{\ifx\empty\rowstblst\else\omit\lineglue\ea\srp\fi
\logmsp3\hfil\prs\lineglue\cr\ifx\empty\rowstblst\else\ea\nxtrs\fi}
\def\caption{{\Large Caption}}
\def\header{\logmsp4 Header\hfil$\Downarrow$\ }
\def\rowstblst{{row 1}{row 2}{row 3}{{\hfil$\Rightarrow$}}}
\def\footer{Footer}
\vcenter{\ruled\framed\btable\data}$$
```

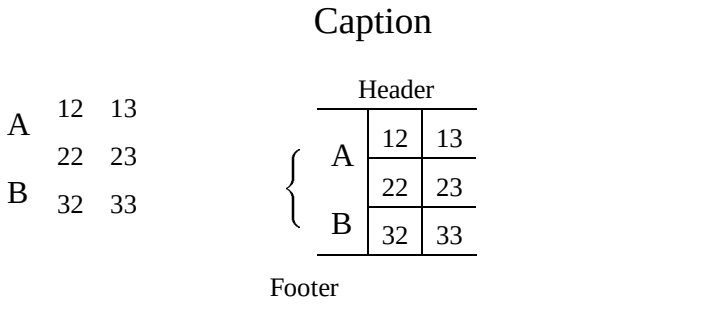
- **Even blocks**, 2-by-1 etc. Leave the cells for the block open in `\data`. Use `\grs`, `\prs`, `\lineglue`, and the like. Define `\ghostrow`.



can be obtained via

```
$$\setbox\block=\hbox{\large A}%2-by-1 block
\def\data{\cs12\cs13\grs
          \cs22\cs23\rs
        31\cs32\cs33}                                          %end \data
\def\ghostrow{\ifx\empty\rowstblst\else\omit\lineglue\ea\srp\fi%
\hfil\vbox to0pt{\vss\copy\block\vss}\hfil\prs
   \logmsp2\lineglue\cr\nxtrs}
{\ruled\framed\setbox0=\btable\data}%for measurement
\vcenter{\dotruled\btable\data} \qquad\qquad
%
\setbox\block=\hbox{\large$2*2$}
\def\data{\logmsp2\hfil\cs13\grs
          \logmsp2\hfil\cs23\rs
                  31\cs32\cs33}                                %end \data
\def\ghostrow{\ifx\empty\rowstblst\else\omit\lineglue\ea\srp\fi%
   \logmsp2\colsepsurround\hfil%
   \vbox to0pt{\copy\block\vss}\hfil\prs\lineglue\cr\nxtrs}
\vcenter{\framed\btable\data}    \qquad\qquad
%
\setbox\block=\hbox{\large$2*3$}
\def\data{\logmsp3\cs14\grs
          \logmsp3\cs24\rs
         31\cs32\cs33\cs34}                                    %end \data
\def\caption{{\Large Caption}}
\def\rowstblst{{row 1}{row 2}{{\hfil$\Rightarrow$}}}
\def\header{\logmsp4 Header\hfil$\Downarrow$\ }
\def\footer{Footer}
\def\ghostrow{\ifx\empty\rowstblst\else\omit\lineglue\ea\srp\fi
\logmsp3\hfil%
\vbox to0pt{\vss\copy\block\vss}\hfil\prs\lineglue\cr\nxtrs}
\vcenter{\framed\ruled\btable\data}$$
```

- **Multiple blocks**: in row stub and table proper



via

```
$$\setbox\block=\vbox{\hbox{\large A}\kern3ex\hbox{\large B}}
\def\data{                              \cs12\cs13\grs
$\vcenter to0pt{\vss\copy\block\vss}$\cs22\cs23\grs
                                        \cs32\cs33\lr}
\vcenter{\btable\data}\qquad\qquad\qquad
```

```
\def\caption{{\Large Caption}}
\def\header{\logmsp3\hfill Header\hfill}
\def\rowstblst{{}{$\vcenter to0pt{\vss\Bigg\{\vss}$}{{}}} $$
\def\footer{Footer}
\vcenter{\ruled\btable\data}
```

with

```
\def\ghostrow{\rss\prs\logmsp2\lineglue\cr\nxtrs}
\def\lr{\tstrut\colsepsurround\cr\prs\logmsp3\lineglue\gobble}
\def\gobble#1#2{}%kludge to undo \tstrut, \colsepsurround
\def\preinsert{\def\rss{&\colsepsurround\hfil\colsepsurround&}
   \def\hs{\colsepsurround\tstrut\cr\prs\logmsp3\lineglue\cr\nxtrs}}
```

Not much is gained in clarity for complicated tables. The logical structure at the top level is there, however.

6 Conclusions

Tables are diverse, and a variety of tools is needed.

Some (deterministic) tables can be generated completely by computer, no detailed user mark up is needed, just the invocation of the macro.

Mark up of bordered scientific tables can be done via the provided bordered table macro `\btable`, in the SGML spirit.

Complex tables need detailed formatting commands and TEXnowledge, in agreement with DEK, TEXbook p.245

> 'People who know how to make ruled tables are generally known as TEX masters. Are you ready?'

Hard things: not to introduce parameters which are already available, and to avoid redundancy. However, I could not get around `\colsepsurround`, because of the impossibility to control the kind of glue inserted by `\tabskip`.
The lack of dotted equivalents of `\hrule`, and `\vrule`, as TEX primitives, is regrettable.

Typesetting tables requires a discipline to be adhered to. What about a discipline of TEXing?

References

[1] ASSOCIATION OF AMERICAN PUBLISHERS, "Markup of Tabular Material", Version 2.0 AAP Inc., 1989.
[2] Asher, "Inside Type & Set", TUGboat, 13.1, 1992, pp. 13–22.
[3] Anne BRÜGGEMAN-KLEIN, D. WOOD, "Drawing Trees nicely with TEX", EP-ODD, 2, 2, 1989, pp.101–115.
[4] David CARLISLE, "longtable.sty", from the fileserver, 1992.
[5] Ray F. COWAN, "tables.tex", from the fileserver, 1985. 7p.
[6] Amy HENDRICKSON, *MacroTEX*, version 0, 1989.
[7] Theo A. JURRIENS, "supertabular.sty", from the fileserver, 1989.
[8] Khanh HA, "Easy Table", TUGboat, 11.2, 1990, pp. 250–264.
[9] Donald E. KNUTH, *The Art of Computer Programming, 3. Sorting and Searching*, Addison-Wesley, Reading Mass., 1973.
[10] Donald E. KNUTH, *The TEXbook*, Addison-Wesley, Reading Mass., 1984.
[11] Hanna KOŁODZIEJSKA, "Go diagrams with TEX", MAPS 91.2, 1991, pp. 63–68.
[12] Kees VAN DER LAAN, "Typesetting Bridge via TEX", TUGboat, 11.2, 1990, pp. 265–276. (An earlier LATEX set-up appeared in TUGboat, 10.1, 1989, pp. 113–116.)
[13] Kees VAN DER LAAN, "SGML(, TEX and . . .)", TUGboat, 12.1, 1991, pp. 90–104. Proceedings EuroTEX 90. Also MAPS 90.1.
[14] Kees VAN DER LAAN, "Math into BLUes, Part I", TUGboat, 12.4, 1991, pp. 485–501. (Part II has appeared in GUTenberg Cahiers 10&11, Proceedings EuroTEX 91.) Also MAPS 90.2.
[15] Kees VAN DER LAAN, "Tower of Hanoi, revisited", TUGboat, 13.1, 1992, pp. 91–94. Also MAPS 92.1, pp. 125–127.
[16] Kees VAN DER LAAN, "Typsetting Crosswords via TEX", This proceedings. Also MAPS 92.1, pp. 128–131.
[17] Kees VAN DER LAAN, "FIFO and LIFO incognito", This proceedings. Also MAPS, 92.1, pp. 121–124.
[18] Kees VAN DER LAAN, "FIFO and LIFO sing the BLUes", MAPS, 92.2.
[19] Leslie LAMPORT, *LATEX user's guide and reference manual*, Addison-Wesley, Reading, Mass., 1986.
[20] M. E. LESK, "Tbl – a program to format tables", UNIX manual, pp. 157–174, 1979.
[21] J.E PITTMAN, "Loopy.TEX", TUGboat, 9.3, 1988, pp. 289–291.
[22] David SALOMON, "Advanced TEX course: Insights & Hindsights," MAPS Special, 1992, 252p.
[23] Michael D. SPIVAK, *LAmSTEX—The Synthesis*. TEXplorators.
[24] Piet TUTELAERS, "A font and a style for typesetting chess using LATEX or TEX", TUGboat, 13.1, 1992, pp. 85–90.
[25] Paul E. S. WORMER, "Young tableaux", from the listserver tex-nl@nic.surfnet.nl.

Syntactic Sugar

Kees van der Laan

Hunzeweg 57,
9893 PB Garnwerd, The Netherlands
cgl@rug.nl

September 1992

Abstract

A plea is made for being honest with TEX and not imposing alien structures upon it, otherwise than via compatible extensions, or via (non-TEX) user interfaces to suit the publisher, the author, or the typist. This will facilitate the process to get (complex) publications out effectively, and typographically of high-quality.

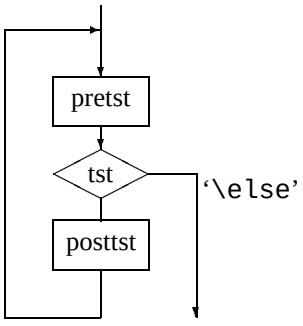
Keywords: (nested) loop, switch, array addressing, keyword and optional parameters, linear search, sorting, plain TEX, macro writing, education.

1 Introduction

TEX is a formatter and also a programming language. TEX is different from current high-level programming languages, but very powerful. A class on its own, and therefore unusual, and unfamiliar.
Because of TEX being different, macro writers propose to harnass TEX into a more familiar system, by imposing syntaxes borrowed from various successful high-level programming languages. In doing so, injustice to TEX's nature might result, and users might become intimidated, because of the difficult—at least unusual—encoding used to achieve the aim. The more so when functional equivalents are already there, although perhaps hidden, and not tagged by familiar names. This is demonstrated with examples about the loop, the switch, array addressing, optional and keyword parameters.
Furthermore, TEX encodings are sometimes peculiar, different from the familiar algorithms, possibly because macro writers are captivated by the mouth processing capabilities of TEX. Users who don't care so much about TEX's programming power, but who are attracted by the typesetting quality, which can be obtained with TEX as formatter, can be led astray when in search for a particular functionality they stumble upon unusual encodings. They might conclude that TEX is too difficult, too error-prone and more things like that and flee towards Word$whatever$, or embrace Desk Top Publishing systems.
The way out is education, next to the provision of compatible, well-documented and supported user interfaces, which don't act like syntactic sugar, by neglecting, or hiding, the already available functional equivalents. Neither the publication of encodings, nor the provision of encodings via file servers or archives—although a nice supporting feature for the TEX-ies—is enough. The quality, compatibility and the simplicity of the (generic) macros should be warranted too.
It is not the aim of this paper to revitalize a programming languages notation war, but to stimulate awareness, and exchange ideas.

2 Loops

DEK's loop, TEXbook p.219, implements the flow



with syntax[1]

`\loop`$\langle pretst\rangle$`\if...`$\langle posttst\rangle$`\repeat`.

Special cases result when either $\langle pretst\rangle$, or $\langle posttst\rangle$ is empty. The former is equivalent to for example PASCAL's **while** . . . **do** . . . , and the latter to **repeat** . . . **until**. With this awareness, I consider the variants as proposed by for example Pittman, [22], and Spivak, [26], as syntactic sugar.
If `\ifcase...` is used, then we have for $\langle posttst\rangle$ several parallel paths, of which one—determined

[1] No (user) '`\else`' is allowed in here, because it is already used in '`\iterate`,' TEXbook p219.

dynamically—will be traversed. Provide and choose your path! What do you mean by traversing the `\else`-path?

2.1 Why another loop?

Kabelschacht, [12], and Spivak, [26], favour a loop which allows the use of `\else`.[2] I have some objections to Kabelschacht's claim that his loop is a *generalization* of plain's loop.

First, it is not a generalization, just a clever, but variant, implementation of the loop flow chart.

Second, it is not compatible with plain's loop. His exit path is via the `\then` branch (or via any of the `\or`-s, when `\ifcase` is used), and not via the `\else` branch.

The reason, I can think of, for introducing another loop, while the most general form has been implemented already, is the existence of commands like `\ifvoid`, and `\ifeof`, and the absence of their negatives `\ifnonvoid`, respectively `\ifnoneof`. In those cases we like to continue the loop via the `\else` branch. For the latter case this means to continue the loop when the file is *not* ended. This can be attained via modifying the loop, of course, but I consider it simpler to use a `\newif` parameter, better known as 'boolean' or 'logical' in other programming languages. With the `\newif` parameter, `\ifneof`, the loop test for an end of file—functionally ¬`\ifeof`—can be obtained via

```
\ifeof\neoffalse\else\neoftrue\fi\ifneof
```

For an example of use, see the Sort It Out subsection. Related to the above encoding of the logical ¬, are the encodings of the logical and, ∧, and or, ∨, via

Functional code	TEX encoding
¬`\if...`	`\if...\notfalse\else \nottrue\fi\ifnot`
`\if...`∧`\if...`	`\andtrue\if...\if... \else\andfalse \else\andfalse\fi\fi \ifand`
`\if...`∨`\if...`	`\ortrue \if...\else\if...\else \orfalse\fi\fi \ifor`

with the `\newif`-s: `\ifnot`, `\ifand`, and `\ifor`.

2.2 Nesting of loops

Pittman, [22], argued that there is a need for other loop encodings.

> 'Recently, I encountered an application that required a set of nested loops and local-only assignments and definitions. TEX's `\loop...\repeat` construction proved to be inadequate because of the requirement that the inner loop be grouped.'

If we take his (multiplication) table—I like to classify these as deterministic tables, because the data as such are not typed in—to be representative, then below a variant encoding is given, which does not need Pittman's double looping. The table is typographically a trifle, but it is all about how the deterministic data are encoded. My approach is to consider it primarily as a table, which it is after all. Within the table the rows and columns are generated, via recursion, and not via the `\loop`. Furthermore, I prefer to treat rules, a frame, a header and row stubs as separate items to be added to the table proper, [17]. The *creation* of local quantities is a general TEX aspect. I too like the idea of a hidden counter, and the next best TEX solution via the local counter. The local versus global creation of counters is a matter of taste, although very convenient now and then. The creation of local quantities is tacitly discouraged by DEK's implementation, because there is no explicit garbage collector implemented and therefore no memory savings can be gained. The only thing that remains is protection against programming mistakes, which is indeed important.

Pittman's table, focused at the essential issue of generating the elements, can be obtained via

```
$$\vbox{\halign{&\ \hfil#\hfil\strut\cr
\rows}}$$
```

with

```
\newcount\rcnt\newcount\ccnt\newcount\tnum
\newcount\mrow\newcount\mcol \mrow2 \mcol3
\def\rows{\global\advance\rcnt1
   \global\ccnt0 \cols\ifnum\rcnt=\mrow\swor
   \fi\rs\rows}
\def\swor#1\rows{\fi\crcr}
\def\cols{\global\advance\ccnt1
   \tnum\rcnt \multiply\tnum\ccnt \the\tnum
   \ifnum\ccnt=\mcol\sloc\fi\cs\cols}
\def\sloc#1\cols{\fi}
\def\rs{\cr}\def\cs{&}
```

The result is

1 2 3
2 4 6

The termination of the recursion is unusual. It is similar to the mechanism used on p.379 of the TEXbook, in the macro `\deleterightmost`. The latter TEXnique is elaborated in [6], and [19].

The above shows how to generate in TEX deterministic tables, where the table entries in other programming languages are generally generated via nested loops. One can apply this to other deterministic math tables—trigonometric tables for example— but then we need more advanced arithmetic facilities in TEX (or inputting the data calculated by other tools), not to mention the

[2] Their loops are equivalent to the general form of the loop with the execution of an extra part after the loop.

appropriate mapping of tables which extend the page boundaries.

For a more complete encoding see Table Diversions, [17]. The idea is that rules and a frame be commanded via `\ruled`, and `\framed`. The header via an appropriate definition of `\header`, $\times$, the indication that we deal with a multiplication table, in `\first`, and the row stubs via definition of the row stub list. All independent and separate from the table proper part.

2.3 Loops and novices

Novice TEXies find DEK's loop unusual, so they sugar it into the more familiar **while**, **repeat**, or **for** constructs, encouraged to do so by exercises as part of courseware. From the functionality viewpoint, there is no need for another loop notation.

With respect to the **for** loop, I personally like the idea of a hidden counter, [15], and [22]. The hidden counter has been used in an *additional* way to plain's loop in for example [15] (via `\preloop` and `\postloop`), and will not be repeated here. This way of doing is a matter of taste, which does not harm, nor hinder, because it is a compatible extension.

And, . . . for the nesting of loops we need scope braces, because of the parameter separator `\repeat`. If braces are omitted, the first `\repeat` is mistaken for the outer one, with the result that the text of the outer loop will *not* become the first `\body`. The good way is, to make the inner `\repeat` invisible at the first loop level, by enclosing the inner loop in braces.

The point I like to get accross is, that there is no real need for another loop encoding. Syntactic sugar, yes.

3 Switches, is there a need?

Apart from the `\ifcase...` construct, TEX seems to lack a multiple branching facility with symbolic names. Fine, [6], introduced therefore

```
\def\fruit#1{\switch\if#1\is
a \apple
b \banana
c \cherry
d \date      \end}
```

I have 2, or rather 3, remarks to the above.
First, the 'switch'-functionality is already there.
Second, Fine's implementation is based upon

> 'It is clear that `\switch` must go through the alternatives one after another, reproducing the test. . . '

Well, . . . going through the alternatives one after another is not necessary.
Third, his example, borrowed from Schwarz, [24], can be solved more elegantly without using a 'switch' or nested `\if`-s at all, as shown below.
The first two aspects are related. Fine's functionality can be obtained via

```
\def\fruit#1{\csname fruit#1\endcsname}
%with
\def\fruita{\apple}
\def\fruitb{\banana} %et cetera
```

With for example `\def\apple{{\bf apple}}`, `\fruit a` yields **apple**.

And what about the 'else' part? Thanks to `\csname`, `\relax` will return when the control sequence has not yet been defined. So, if nothing has to happen we are fine. In the other situations one could define `\def\fruitelse{...}`, and make the else fruits refer to it, for example `\def\fruity{\fruitelse}`, `\def\fruitz{\fruitelse}`, etc. When the set is really uncountable we are in trouble, but I don't know of such situations. And, . . . the five letters 'fruit' are there only to enhance uniqueness of the names.

As example Fine gives the problem, treated by Schwarz, [24], to print vowels in bold face.[3]
The problem can be split into two parts. First, the general part of going character by character through a string, and second, to decide whether the character at hand is a vowel or not.
For the first part use for example, `\dolist`, TEXbook ex11.5, or `\fifo`, [19].

```
\def\fifo#1{\ifx\ofif#1\ofif\fi\process
#1\fifo}          \def\ofif#1\fifo{\fi}
```

For the second part, combine the vowels into a string, `aeiou`, and the problem is reduced to the question $\langle char \rangle \in$ `aeiou`? Earlier, I used the latter approach when searching for a card in a bridge hand, [14].[4] That was well-hidden under several piles of cards, I presume? Anyway, searching for a letter in a string can be based upon `\atest`, TEXbook, p.375, or one might benefit from `\ismember`, p.379. I composed the following

```
\def\loc#1#2{%locate #1 in #2
\def\locate##1#1##2\end{\ifx\empty##2%
\empty\foundfalse\else\foundtrue\fi}
\locate#2.#1\end}     \newif\iffound
```

Then `\fifo Audacious\ofif`
yields **Au**d**a**c**iou**s, with

```
\def\process#1{\uppercase{\loc#1}%
{AEIOU}\iffound{\bf#1}\else#1\fi}
```

[3] A bit misplaced example because the actions in the branches don't differ, except for the non-vowel part.
[4] The macro there was called `\strip`.

Note that en-passant we also accounted for uppercase vowels. By the way, did you figure out why a period—a free symbol—was inserted between the arguments for `\locate`? It is not needed in this example.[5] Due to the period one can test for substrings: $string_1 \in string_2$? Because, $\{string_1 \in string_2\} \wedge \{string_2 \in string_1\} \Rightarrow \{string_1 = string_2\}$, we also have the possibility to test for equality of strings, via `\loc`. Happily, there exists the following straightforward, and TEX-specific, way of testing for equality of strings

```
\def\eq#1#2{\def\st{#1}\def\nd{#2}
\ifx\st\nd\eqtrue\else\eqfalse\fi}
```

For lexicographic comparison, see [18], [19].

4 Array addressing

Related to the switch, or the old computed goto as it was called in FORTRAN, is array addressing. In TEX this can be done via the use of `\csname`. An array element, for example elements identified among others in PASCAL by `a[1]` or `a[apple]`, can be denoted in TEX via the control sequences

```
\csname a1\endcsname
%respectively
\csname aapple\endcsname
```

For practical purposes this accessing, or should we say 'reading,' has to be augmented with macros for writing, as given in [7], and [9]. Writing to an array element can be done via

```
\def\a#1#2{\ea\def\csname a#1%
\endcsname{#2}}\a{1}{Contents}
```

Typesetting (reading) via `\csname a1\endcsname` yields `Contents`, after the above.
The point I like to make is, that 'array addressing'—also called table lookup by some authors—is already there, although unusual and a bit hidden, but, . . . we are used to things like strong type-checking, isn't? Once we can do array addressing we can encode all kind of algorithms, which make use of the array data structure. What about sorting? See the Sort It Out subsection, for a glimpse, and the in depth treatment, [18], with $O(n \log n)$ algorithms, and application to glossary and index sorting.

5 Keyword parameters

In TEX literature the functionality of keyword parameters is heavily used. Some authors impose the syntax known from command languages upon TEX, for examples see [2], or [25]. In my opinion this is syntactic sugar, because of the following rhetorical question. What is essentially the difference between

```
\ref
\key W\by A. Weil
\paper Sur ...
...
\endref
```

as detailed in [25], and for example

```
{\def\key{W}\def\by{A. Weil}
\def\paper{Sur ...}...
\endref}%?
```

The typesetting is done in the cited case by `\ref...\endref`, and in the alternative case by `\endref`. The values for the keys are the background defaults and those temporarily redefined. Note that in both cases the order of the specifications is free and that defaults (empty) are used, for not explicitly specified values.

In my bordered table macro, [17], I could have introduced keyword parameters obeying the command languages syntax. Happily, I refrained from that. I needed several parameters. A parameter for framing, with functionalities nonframed, framed, and dotframed. A parameter for ruling, with functionalities nonruled, ruled, hruled, vruled, and dotruled. And a parameter for positioning of the elements, with functionalities centered, flushed left, and flushed right. (The first element of each enumerated list of values, acting as the default value.)
Furthermore, I decided to provide the user the possibility to *optionally* specify a caption, a header, a rowstub list, or a footer. If any of these is not explicitly specified, then the item will be absent in print too.[6] This resembles optional parameter behaviour, but has been realized by Knuth's parameter mechanism.
In following DEK's approach, I succeeded in keeping the encoding compact, and transparent. I experienced it as simple, direct, and serving extremely-well the purpose.[7]

5.1 Optional parameters

Among others, in LATEX, [20], the mechanism of optional parameters is used. Optional parameters are a special case of keyword parameters. Knuth used optional/keyword parameters abundantly, and called them just parameters, as opposed to arguments of macros. (Think for example of his various parameters and his `\every...`-s.) So it is already there, although in an unusual way.

Another example which illustrates the arbitrariness of the syntax choice with respect to optional/keyword parameters vs. Knuth's parameters is TUGboat's `\twocol` vs. LATEX's `twocolumn` style option.[8]

[5] If omitted the find of 'bb' in 'ab' goes wrong: abbb vs. ab.bb, will be searched.

[6] Another difficulty was to provide a default template, which can be overridden by the user. This was solved by the same approach.

[7] Earlier, I had a similar experience, van der Laan (1990).

[8] LATEX also provides `\twocolumn`.

5.2 Salomon's plain Makeindex

At NTG's 92 spring meeting David Salomon reported about his work in progress about adapting MakeIndex to work with plain. He used optional parameters, with the function as given in the following table

Source	Typeset	
	document	index
`^[abc]`	abc	abc
`^[xyz]{abc}`	abc	xyzabc
`^|abc|`	`abc`	`abc`
`^|\abc|`	`\abc`	`\abc`
`^{\abc}`	replacement text of `\abc`	same
`^[|\abc|!xyz]{}`	nothing	`\abc`, xyz

and combinations thereof.
The same functionality can be obtained via DEK's parameter mechanism. Only one parameter is needed. Let us call this the token variable `\p`. The idea is that the contents of `\p` has to be inserted before the index-entry in the index, and *not* in the text. Some symbols can be given a special meaning, like Salomon did, for example with ! (to denote a subentry).

Salomon's Source	Alternative
`^[abc]`	`^{abc}`
`^[xyz]{abc}`	`{{\p{xyz}^{abc}}}`
`^|\abc|`	`^{|\abc|}`
`^{\abc}`	`^{\abc}`
`^[|\abc|!xyz]{}`	`{\p{|\abc|!xyz}^{}}`

In the above | denotes TUGboat's verbatim delimiter. The macro for ^ has to be adapted accordingly. It is beyond the scope of this paper to work that out in detail. The point I like to make is, that the specification can be done, equally-well, if not simpler, via Knuth's parameter mechanism.

6 Mouth vs. stomach

When one starts with macro writing in TEX one can't get around awareness of TEX's digestive processing. Mouth processing is unusual. For the moment, I consider it as a special kind of built-in pre-processing, an unusual but powerful *generalization* of the elimination of 'dead branches.'[9]

Now and then encoding is published in TUGboat, and other sources as well, which looks difficult, and which does not seem to reflect the familiar algorithms. Sometimes, it has become difficult, because of the strived after processing in the mouth, see for example, [10], [21].[10] The latter author agrees more or less with what is stated above '. . . although the macros are hard to read. . . '.

What puzzles me, are the following questions.

> Why don't authors provide the straightforward TEX encoding, not restricted to mouth processing, as well?
> Why don't they make clear the *need* for mouth processing, or should I say mouth optimization?
> If so, why don't they start with the straightforward encoding and explain the adaptation steps?

Faced with the above questions myself, I would answer that it is apparently too difficult to do so.[11] Furthermore, I read and worked on the Math parts, the alignment parts, the macro chapter, and a substantial part of the dirty tricks Appendix D of the TEXbook, and did find until now only a comment about the *capability* of TEX's mouth processing along with the macro `\deleterightmost`. I know the argument that it is needed within an `\edef`, a `\write...`, and the like. I have heard of that, but from an application point of view, my obvious answer is: Isn't it possible to do the things outside those constructs, equally-well, and pass through the results?

> If authors don't help me out with the above, I consider the encoding as *l'art pour l'art*. Nothing wrong with that, on the contrary.
> The only thing against that is, that it will spread a negative image about TEX encoding, certainly not under the theoretical computer scientists, but under the day-to-day BLUe-type programmers, if not the authors who just use (La)TEX to get their work out, beautifully.

Agreed, Maus referred to the TEXbook, but Jeffrey could have provided a more intelligible solution, and should have refrained from burying his method under a sort of program correctness math. As it is at the moment, it is easier to start from scratch. I experienced that

[9] Knuth might forgive me my ignorance at this point. My brows are raised, when I see published code, restricted to mouth processing, which looks so verbose and unintelligible. I definitely turn my back on it, when the straightforward alternative encoding is familiar, compact, elegant and generic, despite rumour has it that TEX's mouth has the programming power of the Turing machine. As it is, that is something different from let us say literate programming, to indicate a broad stream of readable programs, in my opinion.

[10] By the way, when do we know that something is completely processed in the mouth? Is there a check on it? Or, . . . is it just an abstract part of the TEXnigma?

[11] And what about the efficiencies? From the viewpoint of the machine and with respect to human understanding? I have not seen the common and mouth versions of an algorithm published simultaneously, let alone have them compared with respect to timing.

already with the encoding of: the Tower of Hanoi, typesetting crosswords, generating n-copies, lexicographis comparison, and sorting. The published encodings inspired me to develop alternatives, that is true, but that should not be the aim, should it? Furthermore, I wonder how many *users* have been discouraged by those 'difficult to read' codes, especially when the familiar codes are straightforward?

6.1 n-copies

I needed Maus' functionality—avant la lettre—in typesetting a fill-in form, where a number of rows had to be repeated. Of course, my editor can do it—statically—and that served the purpose. It is easy for sure, but it does not look elegant. A straightforward use of tail recursion satisfied me better, because of the simplicity, the compactness and the elegance, at the expense of a negligible efficiency loss. See the example about the birdge form in Table Diversions, [17].[12] The tail recursion determines the number of copies dynamically, as do the other solutions given by DEK, for example the nice solution via the use of `\aftergroup`, TEXbook, p.374.

6.2 Sort it out

Jeffrey's problem is: given an unsorted list of (positive) integers via symbolic names, typeset the ordered list. In order to concentrate on the main issues, assume that his list adheres to Knuth's list structure, TEXbook, p.378. As example consider the list[13]

```
\def\lst{\\\ia\\\ib\\\ic}
\def\ia{314}\def\ib{27}\def\ic{1}
```

The sorted numbers 1, 27, 314, are obtained via

```
\def\\#1{\ifnum#1<\min\let\min=#1\fi}
\def\first#1{\def\lop\\##1##2\pol{%
       \let\min=##1}\ea\lop#1\pol}
\newif\ifnoe
\loop\ifx\empty\lst\noefalse\else
     \noetrue\fi
\ifnoe  \first\lst   \lst   \min,
   {\def\\##1{\ifx##1\min\else\noexpand\\
   \noexpand##1\fi}\xdef\lst{\lst}}
\repeat
```

The encoding implements the looping of the basic steps

- find minimum (via `\lst`, and suitable definition of the active list separator `\\`)
- typeset minimum (via `\min`)
- delete minimum from the list (again via an(other) appropriate definition of the active list separator).

For removing a typesetted element, I was inspired by `\remequivalent`, TEXbook, p.380.[14]

The above is effective for short lists, as was the case in Jeffrey's application.[15] For longer lists, techniques of order O($n \log n$) are more appropriate. For plain TEX encodings see [18].

6.3 Lexicographic comparison

Eijkhout, [5], provided macros—focused at mouth processing—for lexicographic ordering.
His `\ifallchars...\are...\before` made ample use of `\expandafter`, and is not easy accessible for somebody with say 2 years of TEX experience. Hackers might go into ecstasy, but application oriented users become discouraged. For a straightforward alternative, not restricted to mouth processing, see [19]. The point I like to make is, that I would have welcomed the familiar solution and the transformation steps as well.

7 Acknowledgements

Włodek Bzyl and Nelson Beebe are kindly acknowledged for their help in clearing up the contents and correcting my use of English, respectively.

8 Conclusion

It is hoped that authors who can't resist the challenge to impose syntaxes from successful programming languages upon TEX, also encode the desired functionality in TEX's peculiar way, and contrast this with their proposed improvements. The novice, the layman and his peers will benefit from it.

The difficulties caused by TEX's unusual encoding mechanisms, can best be solved via education, and not via imposing structures from other languages. The latter will entail confusion, because of all those varieties. Furthermore, it is opposed to the Reduced Instruction Set idea, which I like. For me it is similar to the axioms-and-theorems structure in math, with a minimal number of axions, all mutual orthogonal.

Publishing houses, user groups, and macro writers are encouraged to develop and maintain 'user interfaces,' which do justice to TEX's nature, and don't increase the complexity of TEX's components. Good examples are: TUGboat's sty files, AMS-LATEX & $\mathcal{AMS}$-TEX, and L$\mathcal{AMS}$-TEX. Macro-TEX and lxiii[16] are promising.

[12] The complexity is of order O(n), instead of O($\log n$), which is not important, because of the small number of copies involved.

[13] Equally-well, the comma could have been used as an active list separator, which looks more natural. I decided to adhere to Knuth's notation.

[14] I was not able to apply the parameter separator technique to locate the element to be removed.

[15] Remember, that sorting based on linear search has complexity O(n^2).

[16] The LATEX3 project.

File servers and archives are welcomed, but the compatibility, the simplicity and in general the quality, must be warranted too. Not to mention pleasant documentation and up-to-date-ness.
My wishful thinking is to have intelligent local archives, which have in store what is locally generally needed, and know about what is available elsewhere. The delivery should be transparent, and independent whether it comes from elsewhere or was in store.

References

[1] Appelt, W (1987): Macros with keyword parameters. *TUGboat* 8, no. (2), 182–184.
[2] Appelt, W (1988): TEX für Fortgeschrittene, Programmiertechniken und Makropakete. Addison-Wesley.
[3] Beebe, N.H.F (1991): The TUGlib server. MAPS 91.2, 117–123.
[4] Beeton, B.N, R. Whitney (1989): *TUGboat* 10, no. (3), 378–385.
[5] Eijkhout, V (1991): TEX by Topic. Addison-Wesley.
[6] Fine, J (1992): Some basic control macros for TEX. *TUGboat* 13, no. (1), 75–83.
[7] Greene, A. M (1989): TEXreation—Playing games with TEX's mind. TUG89. *TUGboat* 10, no. (4), 691–705.
[8] Hendrickson, A (1989): MacroTEX.
[9] Hendrickson, A (1990): Getting TEXnical: Insights into TEX macro writing techniques. *TUGboat* 11, no. (3), 359–370.
[10] Jeffreys, A (1990): Lists in TEX's mouth. *TUGboat* 11, no. (2), 237–244.
[11] Jensen, K, N. Wirth (1975): PASCAL user manual and report. Springer-Verlag. *TUGboat* 11, no. (2), 237–244.
[12] Kabelschacht, A (1987): `\expandafter` vs. `\let` and `\def` in conditionals and a generalization of plain's `\loop`. *TUGboat* 8, no. (2), 184–185.
[13] Knuth, D.E (1984): *The TEXbook*, Addison-Wesley.
[14] Laan, C.G van der (1990): Typesetting Bridge via TEX. *TUGboat* 11, no. (2), 265–276.
[15] Laan, C.G van der (1992a): Tower of Hanoi, revisited. *TUGboat* 13, no. (1), 91–94.
[16] Laan, C.G van der (1992b): FIFO & LIFO incognito. EuroTEX '92, 225–234. Also MAPS92.1. An elaborated version is FIFO & LIFO sing the BLUes.
[17] Laan, C.G van der (1992c): Table Diversions. EuroTEX '92, 191–211. A little adapted in MAPS92.2.
[18] Laan, C.G van der (in progress): Sorting in BLUe. MAPS93.1. Heap sort encoding is released in MAPS92.2.
[19] Laan, C.G van der (1992d): FIFO & LIFO sing the BLUes. MAPS92.2.
[20] Lamport, L (1986): LATEX, user's guide & reference manual. Addison-Wesley.
[21] Maus, S (1991): An expansion power lemma. *TUGboat* 12, no. (2), 277.
[22] Pittman, J.E (1988): Loopy.TEX. *TUGboat* 9, no. (3), 289–291.
[23] Salomon, D (1992): NTG's Advanced TEX course: Insights & Hindsights. MAPS 92 Special. 254p.
[24] Schwarz, N (1987): *Einführung in TEX*, Addison-Wesley.
[25] Siebenmann, L (1992): Elementary Text Processing and Parsing in TEX. *TUGboat* 13, no. (1), 62–73.
[26] Spivak, M.D (1987): LAMSTEX. TEXplorators.
[27] Youngen, R.E (1992): TEX-based production at AMS. MAPS92.2. 7p.

Heap Sort in TEX

Kees van der Laan

Hunzeweg 57,
9893 PB Garnwerd, The Netherlands
cgl@rug.nl

September 1992

Abstract

Sorting in plain TEX is implemented via heap sort. The heap sort algorithm is explained and the encoding given.

Keywords: (heap) sort, array addressing, plain TEX, macro writing, education.

1 Introduction

A TEX encoding of the heap sort algorithm is given. A simpler user interface and examples of use, also with respect to lexicographic sorting, will be treated in Sorting in BLUe, to come.

2 Example of use

```
\ea\def\csname1\endcsname{27}
\ea\def\csname2\endcsname{1}
\ea\def\csname3\endcsname{314} \n=3
\heapsort
\csname1\endcsname,
\csname2\endcsname,
\csname3\endcsname.
```

yields 314, 27, 1.

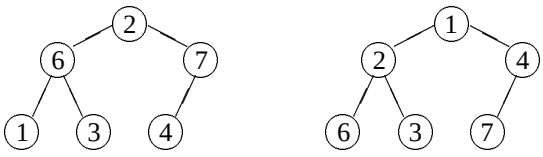
3 Heap sort

The process consist of two main steps

- creation of a heap
- sorting the heap

with a sift operation to be used in both.

What is a heap? A sequence $a_1, a_2, \ldots, a_n$, is a heap if $a_k \leq a_{2k} \wedge a_k \leq a_{2k+1}, k = 1, 2, \ldots, n \div 2$, and because a_{n+1} is undefined, it is defined that $a_k < a_{n+1}$ is true. A tree and one of its heap representations of $2, 6, 7, 1, 3, 4$, is displayed below



3.1 The algorithm

In PASCAL-like notation the algoritm reads
%heap creation in '1:n'
$l := n$ **div** $2 + 1;$
while $l \neq 1$ **do** $l := l - 1; sift(a, l, n)$ **od**
%sorting in '1:n'
$r := n;$ **while** $r \neq 1$ **do**
$exchange(a[1], a[r]) r := r - 1; sift(a, 1, r)$ **od**
%sift #1 through #2
$j := \#1$
while $2j \leq \#2 \wedge (a[j] > a[2j] \vee a[j] > a[2j + 1])$ **do**
$mi := 2j +$ **if** $a[2j] < a[2j + 1]$ **then** 0 **else** 1 **fi**
$exchange(a[j], a[mi]) j := mi$ **od**

3.2 TEX encoding

```
\let\ea=\expandafter          %28/9/92
\newcount\n\newcount\l\newcount\r
\newcount\i\newcount\uone
\newcount\j\newcount\jj\newcount\jjone
\newif\ifcmp\newif\ifgoon \newif\iflt
\def\heapsort{%data in \1 to \n
\r=\n \heap \i=1 %
\loop\ifnum\r>1 \exchange\i\r%
   \advance\r by-1 {\sift\i\r}%
\repeat}
%
\def\heap{%Transform \1..\n into heap
\l=\n\divide\l by2 \advance\l by1 %
\loop\ifnum\l>1 %
   \advance\l by-1 {\sift\l\n}%
\repeat}
%
\def\cmpval#1#2{%#1, #2 counter variables
\xdef\aone{\csname\the#1\endcsname}%
\xdef\atwo{\csname\the#2\endcsname}%
\cmptrue\ifnum\aone>\atwo{}\cmpfalse\fi}
%
\def\exchange#1#2{%#1, #2 counter variables
\edef\aux{\csname\the#1\endcsname}%
\ea\xdef\csname\the#1\endcsname{\csname
\the#2\endcsname}%
```

```
\ea\xdef\csname\the#2\endcsname{\aux}}
%
\def\sift#1#2{%#1, #2 counter variables
\jj=#1 \uone=#2 \advance\uone1 \goontrue%
\loop \j=\jj \advance\jj by\jj%
 \ifnum\jj<\uone%
   \jjone=\jj \advance\jjone by1%
   \ifnum\jj<#2{}\cmpval\jj\jjone%
         \ifcmp\else\jj=\jjone\fi\fi%
   \cmpval\j\jj\ifcmp\goonfalse\fi%
 \else\goonfalse\fi%
\ifgoon\exchange\j\jj\repeat}
```

heapsort
The values of `\1`,...,`\n`, are sorted in descending order: `\1` $\geq$ `\2` $\geq \cdots \geq$ `\n`.

heap
The values `\1`,...,`\n`, are rearranged into a heap.

sift
The first element denoted by the first (counter) argument has disturbed the heap, because of the exchange of the first and the last (one higher than the second argument) element. **sift** sifts this first element through the heap until the heap property holds again.

FIFO and LIFO sing the BLUes*

Kees van der Laan

Hunzeweg 57,
9893 PB Garnwerd, The Netherlands
cgl@rug.nl

September 1992

Abstract

FIFO, First-In-First-Out, and LIFO, Last-In-First-Out, are well-known techniques for handling sequences. In TEX macro writing they are abundant but are not easily recognized as such. TEX templates for FIFO and LIFO are given and their use illustrated. The relation with Knuth's `\dolist`, answer ex11.5, and `\ctest`, p.376, is given.

Keywords: FIFO, LIFO, list processing, plain TEX, education, macro writing.

1 Introduction

It started with the programming of the Tower of Hanoi in TEX, van der Laan (1992a). For printing each tower the general FIFO—First-In-First-Out[1]—approach was considered.[2] In literature (and courseware) the programming of these kind of things is done differently by each author, inhibiting intelligibility. In pursuit of Wirth (1976), TEX templates for the FIFO (and LIFO) paradigm will hopefully improve the situation.

2 FIFO

In the sequel, I will restrict the meaning of FIFO to an input stream which is processed argument-wise. FIFO can be programmed in TEX as template

```
\def\fifo#1{\ifx\ofif#1\ofif\fi\process
#1\fifo}     \def\ofif#1\fifo{\fi}
```

Printing of a tower can be done via

```
\def\process#1{\hbox to3ex{%
\hss\vrule width#1ex height1ex\hss}}
\vbox{\baselineskip1.1ex\fifo12\ofif}
```

For the termination of the tail recursion the same TEXnique as given in the TEXbook, p.379, in the macro `\deleterightmost`, is used. This is elaborated as `\break` in Fine (1992), in relation to termination of the loop. The idea is that when `\ofif` is encountered in the input stream, all tokens in the macro up to and including `\fifo`—the start for the next level of recursion—are gobbled.[3] Because the matching `\fi` is gobbled too, this token is inserted via the replacement text of `\ofif`. This TEXnique is better than Kabelschacht's, (1987), where the token preceding the `\fi` is expanded after the `\fi` via the use of `\expandafter`. When this is applied the exchange occurs at each level in the recursion. It also better than the `\let\nxt=...` TEXnique, which is used in the TEXbook, for example in `\iterate`, p.219, because there are no assignments.

My first version had the two tokens after `\ifx` reversed—a cow flew by—and made me realize the non-commutativity of the *first level* arguments of TEX's conditionals. For example, `\ifx aa\empty...` differs from `\ifx\empty aa...`, and `\if\ab\aa...` from `\if\aa\ab...`, with `\def\aa{aa}`, `\def\ab{ab}`. In math, and in programming languages like PASCAL, the equality relation is commutative,[4] and no such thing as expansion comes in between. When not alert with respect to expansion, TEX's `\if`-s can surprise you.

The `\fifo` macro is a basic one. It allows one to proceed along a list—at least conceptually—and to apply a (user) specified process to each list element. By this approach the programming of going through a list is *separated* from the various processes to be applied to

*Earlier versions appeared in MAPS92.1 and proceedings EuroTEX '92.

[1] See Knuth (1968), section 2.2.1.

[2] In the Tower of Hanoi article Knuth's list datastructure was finally used—TEXbook Appendix D.2—with FIFO inherent.

[3] In contrast with usual programming of the recursion start with the infinite loop, and then insert the `\if...\ofif\fi`.

[4] So are TEX's `\if`-s after expansion.

the elements.[5] It adheres to the *separation of concerns* principle, which I consider fundamental.

The input stream is processed argument-wise, with the consequence that first level braces will be gobbled. Beware! Furthermore, no outer control sequences are allowed, nor \par-s. The latter can be permitted via the use of \long\def.

A general approach—relieved from the restrictions on the input stream: *every token* is processed until \ofif—is given in the TEXbook answer ex11.5 (\dolist...) and on p.376 (\ctest...). After adaptation to the \fifo notation and to the use of macros instead of token variables, Knuth's \dolist comes down to

```
\def\fifo{\afterassignment\tap
\let\nxt= }
\def\tap{\ifx\nxt\ofif\ofif\fi\process
\nxt\fifo}       \def\ofif#1\fifo{\fi}
```

This general approach is indispensable for macro writers. My less general approach can do a lot already, for particular applications, as will be shown below. But, . . . beware of its limitations.

2.1 Variations

The above \fifo can be seen as a template for encoding tail recursion in TEX, with arguments taken from the input stream one after another. An extension is to take two arguments from the input stream at a time, with the second argument to look ahead, via

```
\def\fifo#1#2{\process#1\ifx\ofif#2
    \ofif\fi\fifo#2}
\def\ofif#1\ofif{\fi}
```

Note the systematics in the use of the parameter separator in \ofif.

And what about recursion without parameters? A nice example of that is a variant implementation of Knuth's \iterate of the \loop, TEXbook, p.219

```
\def\iterate{\body\else\etareti\fi%
\iterate}   \def\etareti#1\iterate{\fi}
```

(This \iterate contains only 5 tokens in contrast with Knuth's 11. The efficiency and the needed memory is determined by the number of tokens in \body, and therefore this 5 vs. 11 is not relevant.)

2.2 Variable number of parameters

TEX macros can take at most 9 parameters. The above \fifo macro can be seen as a macro which is relieved from that restriction. Every group, or admissible token, in the input stream after \fifo up to and including \ofif, will become an argument to the macro. When the \ofif token is reached, the recursion will be terminated.[6]

2.3 Unknown number of arguments

Tutelaers (1992), as mentioned by Eijkhout (1991), faced the problem of inputting a chess position. The problem is characterized by an unspecified number of positions of pieces, with for the pawn positions the identification of the pawn generally omitted. Let us denote the pieces by the capital letters K(ing), Q(ueen), B(ishop), (k)N(ight), R(ook), and P(awn), with the latter symbol default. The position on the board is indicated by a letter a, b, c, . . . , or h, followed by a number, 1, 2, . . . , or 8. Then, for example,

```
\position{Ke1, Qd1, Na1, e2, e4}
```

should entail the invocations

```
\piece{K}{e1}\piece{Q}{d1}\piece{N}{a1}
\piece{P}{e2}\piece{P}{e4}
```

This can be done by an appropriate definition of \position, and an adaptation of the \fifo template, via

```
\def\position#1{\fifo#1,\ofif,}
\def\fifo#1,{\ifx\ofif#1\ofif
\fi\process#1\relax\fifo}
\def\ofif#1\fifo{\fi}
\def\process#1#2#3{\ifx\relax#3
\piece{P}{#1#2}\else\piece#1{#2#3}\fi}
```

With the following definition (simplified in relation to Tutelaers)

```
\def\piece#1#2{ #1-#2}
```

we get K-e1 Q-d1 N-a1 P-e2 P-e4.

For an unknown number of arguments at two levels see the Nested FIFO section.

2.4 Length of string

An alternative to Knuth's macro \getlength, TEXbook p.219, is obtained via the use of \fifo with

```
\newcount\length
\def\process#1{\advance\length1 }
```

Then `\fifo aap  noot\ofif\number\length`

yields the length 7.[7]

[5] If a list has to be *created*, Knuth's list datastructure might be used, however, simplifying the execution of the list. See TEXbook Appendix D.2.

[6] Another way to circumvent the 9 parameters limitation is to associate names to the quantities to be used as arguments, let us say via def's, and to use these quantities via their names in the macro. This is Knuth's parameter mechanism and is functionally related to the so-called keyword parameter mechanism of command languages, and for example ADA.

[7] Insert \obeyspaces when the spaces should be counted as well.

2.5 Number of asterisks

An alternative to Knuth's `\atest`, TEXbook, p.375, for determining the number of asterisks, is obtained via `\fifo` with

```
\def\process#1{\if*#1\advance\acnt by1
\fi}\newcount\acnt
```

Then `\fifo abc*de*\ofif \number\acnt` yields the number of asterisks: 2.[8]

2.6 Vertical printing

David Salomon treats the problem of vertical printing in his courseware. Via an appropriate definition of `\process` and a suitable invocation of `\fifo` it is easily obtained.

```
\def\process#1{\hbox{#1}}
xy\vbox{\offinterlineskip\fifo abc\ofif}yx
```

yields xy$\begin{matrix}a\\b\end{matrix}$yx.

2.7 Delete last character of argument

Again an example due to David Salomon. It is related to `\deleterightmost`, TEXbook p.379. Effective is the following, where a second parameter for `\fifo` is introduced to look ahead, which is inserted back when starting the next recursion level

```
\def\gobblelast#1{\fifo#1\ofif}
\def\fifo#1#2{\ifx\ofif#2\ofif
   \fi#1\fifo#2}
\def\ofif#1\ofif{\fi}
```

Then `\gobblelast{aap}` will yield aa.

2.8 Vowels, voilà

Schwarz (1987) coined the problem to print vowels in bold face.[9] The problem can be split into two parts. First, the general part of going character by character through a string, and second, decide whether the character at hand is a vowel or not.

For the first part use `\fifo` (or Knuth's `\dolist`). For the second part, combine the vowels into a string, `aeiou`, and the problem can be reduced to the question $\langle char \rangle \in$ `aeiou`? Earlier, I used this approach in searching a card in a bridge hand, van der Laan (1990, the macro `\strip`). That was well-hidden under several piles of cards, I presume? The following encoding is related to `\ismember`, TEXbook, p.379

```
\newif\iffound
\def\loc#1#2{%locate #1 in #2
\def\locate##1#1##2\end{\ifx\empty##2%
\empty\foundfalse\else\foundtrue\fi}%
\locate#2#1\end}
```

Then `\fifo Audacious\ofif` yields **Au**d**a**c**iou**s, with

```
\def\process#1{\uppercase{\loc#1}%
{AEIOU}\iffound{\bf#1}\else#1\fi}
```

2.9 Variation

If in the invocation `\locate#2#1` a free symbol is inserted between `#2` and `#1`, then `\loc` can be used to locate substrings.[10] And because $\{string_1 \in string_2\} \wedge \{string_2 \in string_1\} \Rightarrow string_1 = string_2$, the variant can be used for the equality test for strings. See also the Multiple FIFO subsection, for general and more effective alternatives for equality tests of strings.

2.10 Processing lines

What about processing lines of text? In official, judicial, documents it is a habit to fill out lines of text with dots.[11] This can be solved by making the end-of-line character active, with the function to fill up the line. A general approach where we can `\process` the line, and not only append to it, can be based upon `\fifo`.

One can wonder, whether the purpose can't be better attained by filling up the last line of paragraphs by dots, because TEX justifies with paragraphs as units.

2.11 Processing words

What about handling a list of words? This can be achieved by modifying the `\fifo` template into a version which picks up words, `\fifow`, and to give `\processw` an appropriate function.

```
\def\fifow#1 {\ifx\ofifw#1\ofifw\fi
    \processw{#1}\ \fifow}
\def\ofifw#1\fifow{\fi}
```

2.12 Underlining words

In print it is uncommon to emphasize words by underlining. Generally another font is used, see discussion of exercise 18.26 in the TEXbook. However, now and then people ask for (poor man's) underlining of words. The following `\processw` definition underlines words picked up by `\fifow`

```
\def\processw#1{\vtop{\hbox{\strut#1}
                \hrule}}
```

[8] As the reader should realize, this works correctly when there are first level asterisks *only*. For counting at all levels automatically, a more general approach is needed, see Knuth's `\ctest`, p.376.

[9] His solution mixes up the picking up of list elements and the process to be applied. Moreover, his nesting of `\if`-s in order to determine whether a character is a vowel or not, is not elegant. Fine (1992)'s solution, via a switch, is not elegant either.

[10] Think of finding 'bb' in 'ab' for example, which goes wrong without the extra symbol.

[11] The problem was posed at EuroTEX '91 by Theo Jurriens.

Then

```
\leavevmode\fifow leentje leerde lotje
lopen langs de lange lindenlaan \ofifw
\unskip.
```

yields leentje leerde lotje lopen langs de lange lindenlaan.

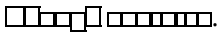
3 Nested FIFO

One can nest the FIFO paradigm. For processing lines word by word, or words character by character.

3.1 Words character by character

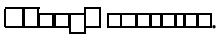
Ex11.5, can be solved by processing words character by character. A solution to a slightly simplified version of the exercise reads

```
\fifow Though  exercise \ofifw \unskip.
%with
\def\processw#1{\fifo#1\ofif}
\def\process#1{\boxit#1}
\def\boxit#1{\setbox0=\hbox{#1}\hbox
{\lower\dp0\vbox{\offinterlineskip\hrule
\hbox{\vrule\phantom#1\vrule}\hrule}}}
```

yields  .

In the spirit of `\dolist...`, ex11.5, is

```
%variant neglecting word structure
\def\fifo{\afterassignment\tap
\let\nxt= }
\def\tap{\ifx\nxt\ofif\ofif
    \fi\process\nxt\fifo}
\def\ofif#1\fifo{\fi}
\def\process#1{\if\space\nxt\
\else\boxit#1\fi}
\fifo Though    exercise\ofif.
```

with the same result  .

3.2 Mark up natural data

Data for `\h(v)align` needs & and `\cr` marks. We can get plain TEX to append a `\cr` at each (natural) input line, TEXbook p.249. An extension of this is to get plain TEX to insert `\cs`-s, column separators, and `\rs`-s, row separators, and eventually to add `\lr`, last row, at the end, in natural data. For example prior to an invocation of `\halign`, one wants to get plain TEX to do the transformation

P*ON
DEK* ⇒ P\cs*\csO\csN\rsD\csE\csK\cs*\lr

This can be done via

```
$$\vcenter{\hbox{P*ON}\kern.5ex
  \hbox{DEK*}} \,\Rightarrow\,
%And now right, mark up part
\bdata P*ON
DEK*
\edata\markup\data
\vcenter{\hbox{\data}}$$
```

with

```
\def\bdata{\bgroup\obeylines\store}
\def\store#1\edata{\egroup\def\data{#1}}
\def\markup#1{\ea\xdef\ea#1\ea{\ea
              \fifol#1\ofifl}}
```

and auxiliaries

```
\let\nx=\noexpand
{\catcode`\^^M=13
\gdef\fifol#1^^M#2{\fifo#1\ofif%
   \ifx\ofifl#2\nx\lr\ofifl
   \fi\nx\rs\fifol#2}}
\def\ofifl#1\ofifl{\fi}
\def\fifo#1#2{#1\ifx\ofif#2\ofif
    \fi\nx\cs\fifo#2}
\def\ofif#1\ofif{\fi}
%with for this example
\def\cs{{\sevenrm{\tt\char92}cs}}
\def\rs{{\sevenrm{\tt\char92}rs}}
\def\lr{{\sevenrm{\tt\char92}lr}}
```

The above came to mind when typesetting crosswords,[12] van der Laan (1992b), while striving after the possibility to allow natural input, independent of `\halign` processing.

4 Multiple FIFO

What about FIFO for more than one stream? (For simplicity the streams are stored in def-s, because `\read` inputs lines.) For example comparing strings, either for equality or with respect to lexicographic ordering? Eijkhout (1992, p.137, 138) provided for these applications the macros

`\ifAllChars...\Are...\TheSame`,

and

`\ifallchars...\are...\bfore`.

The encodings are focused at mouth processing. The latter contains many `\expandafter`-s.

A basic approach is: loop through the strings character by character, and compare the characters until either the assumed condition is no longer true, or the end of either one of the strings, has been reached.

[12] With *, or ⊔, given an appropriate function.

4.1 Equality of strings

The TEX-specific encoding, where use has been made of the property of `\ifx` for control sequences, reads

```
\def\eq#1#2{\def\st{#1}\def\nd{#2}
\ifx\st\nd\eqtrue\else\eqfalse\fi}
```

with auxiliary `\newif\ifeq`.

As a stepping stone for lexicographic comparison, consider the general encoding

```
\def\eq#1#2{\continuetrue\eqtrue
\loop\ifx#1\empty\continuefalse\fi
     \ifx#2\empty\continuefalse\fi
\ifcontinue \nxte#1\nxtt \nxte#2\nxtu
     \ifx\nxtt\nxtu
     \else\eqfalse\continuefalse\fi
\repeat
\ifx\empty#1\ifx\empty#2
\else\eqfalse\fi\else\eqfalse\fi}
```

with auxiliaries

```
\newif\ifcontinue\newif\ifeq
\def\nxte#1#2{\def\pop##1##2\pop{%
\gdef#1{##2}\gdef#2{##1}}\ea\pop#1\pop}
```

Then

```
\def\t{abc}\def\u{ab}
\eq\t\u\ifeq$abc=ab$\else$abc\not=ab$\fi
```

yields $abc \neq ab$.

4.2 Lexicographic comparison

Assume that we deal with lower case and upper case letters only. The encoding of `\sle`—String Less or Equal—follows the same flow as the equality test, `\eq`, but differs in the test, because of TEX's expansion mechanisms

```
\def\sle#1#2{%#1, #2 are def's
\global\sletrue {\continuetrue
\loop\ifx#1\empty\continuefalse\fi
     \ifx#2\empty\continuefalse\fi
\ifcontinue\nxte#1\nxtt\nxte#2\nxtu
     \ea\ea\ea\lle\ea\nxtt\nxtu
\repeat}
\ifsle\ifx\empty#2\ifx\empty#1
      \else\global\sltfalse\fi\fi
\fi}
```

with auxiliaries (lle=Letter Less or Equal)

```
\newif\ifcontinue\global\newif\ifsle
\def\nxte#1#2{\def\pop##1##2\pop{%
\xdef#1{##2}\xdef#2{##1}}\ea\pop#1\pop}
\def\lle#1#2{\uppercase{\ifnum'#1='#2}
\else\continuefalse
     \uppercase{\ifnum'#1>'#2}{}\global
                      \slefalse\fi
\fi}
```

For example

```
\def\t{ABC}\def\u{ab}\sle\t\u
\ifsle$ABC<ab$\else$ABC>ab$\fi
```

yields $ABC > ab$,
and

```
\def\t{noo}\def\u{apen}\sle\t\u
\ifsle$noo<apen$\else$noo>apen$\fi
```

yields $noo > apen$.

The above can be elaborated with respect to `\read` for strings each on a separate file, to strings with accented letters, to the inclusion of an ordering table, and in general to sorting. Some of the mentioned items will be treated in Sorting in BLUe.

5 LIFO

A modification of the `\fifo` macro—`\process{#1}` invoked at the end instead of at the beginning—will yield the Last-In-First-Out template. Of course LIFO can be applied to reversion 'on the fly,' without explicitly allocating auxiliary storage.[13]

```
\def\lifo#1#2\ofil{\ifx\empty#2
  \empty\ofil\fi\lifo#2\ofil\process#1}
\def\ofil#1\ofil{\fi}
```

With the identity—`\def\process#1{#1}`, or the invoke `\process#1` replaced by `#1`[14]—the template can be used for reversion on the fly For example `\lifo aap\ofil` yields `paa`.

5.1 Change of radix

In the TEXbook a LIFO exercise is provided at p.219: print the digits of a number in radix 16 representation. The encoding is based upon the property

$$(N \div r^k) \bmod r = d_k, \quad k = 0, 1, \ldots, n,$$

with radix r, coefficients d_k, and the number representation

$$N = \sum_{k=0}^{n} d_k \, r^k.$$

[13] Johannes Braams drew my attention to Knuth and MacKay (1987), which contained among others `\reflect...\tcelfer`. They compare #1 with `\empty`, which is nice. The invocation needs an extra token, `\empty`—a so-called sentinel, see Wirth (1976)—to be included before `\tcelfer`, however. (Knuth and Mackay hide this by another macro which invokes `\reflect...\empty\tcelfer`). My approach requires at least one argument, with the consequence that the empty case must be treated separately, or a sentinel must be appended after all.

[14] Remember the stack size limitations.

There are two ways of generating the numbers d_k: starting with d_n, or the simpler one starting with d_0, with the disadvantage that the numbers are generated in reverse order with respect to printing. The latter approach is given in TEXbook p.219. Adaptation of the LIFO template does not provide a solution much different from Knuth's, because the numbers to be typeset are generated in the recursion and not available in the input stream.

6 Acknowledgements

Włodek Bzyl and Nelson Beebe are kindly acknowledged for their help in clearing up the contents and correcting my use of English, respectively.

7 Conclusion

In looking for a fundamental approach to process elements sequentially—not to confuse with list processing where the list is also built up, see TEXbook Appendix D.2, or with processing of *every* token in the input stream, see ex11.5 or p.376—TEX templates for FIFO and LIFO, emerged.

The templates can be used for processing lines, words or characters. Also processing of words line by line, or characters word by word, can be handled via nested use of the FIFO principle.

The FIFO principle along with the look ahead mechanism is applied to molding natural data into representations required by subsequent TEX processing.

Courseware might benefit from the FIFO approach to unify answers of the exercises of the macro chapter.

TEX's `\ifx...` and `\if...` conditionals are non-commutative with respect to their *first level* operands, while the similar mathematical operations are, as are the operations in current high-level programming languages.

Multiple FIFO, by comparing strings lexicographically, has been touched upon.

References

[1] Eijkhout, V (1991): TEX by Topic. Addison-Wesley.

[2] Fine, J (1992): Some basic control macros for TEX, *TUGboat* 13, no. (1), 75–83.

[3] Hendrickson, A (priv. comm.)

[4] Kabelschacht, A (1987): `\expandafter` vs. `\let` and `\def` in conditionals and a generalization of plain's `\loop`. *TUGboat* 8, no. (2), 184–185.

[5] Knuth, D.E (1968): The Art of Computer Programming. 1. Fundamental Algorithms. Addison-Wesley.

[6] Knuth, D.E (1984): The TEXbook. Addison-Wesley.

[7] Knuth, D.E, P. Mackay (1987): Mixing right-to-left texts with left-to-right texts. *TUGboat* 7, no. (1), 14–25.

[8] Laan, C.G van der (1990): Typesetting Bridge via TEX, *TUGboat* 11, no. (2), 91–94.

[9] Laan, C.G van der (1992a): Tower of Hanoi, revisited. *TUGboat* 13, no. (1), 91–94.

[10] Laan, C.G van der (1992b): Typesetting Crosswords via TEX. EuroTEX '92, 217–224. Also MAPS92.1.

[11] Laan, C.G van der (1992c): Table Diversions. EuroTEX '92, 191–211. Also a little adapted in MAPS92.2.

[12] Laan, C.G van der (in progress): Sorting in BLUe. MAPS93.1. (For heap sort encoding in plain TEX, see MAPS92.2)

[13] Salomon, D (1992): Advanced TEX course: Insights & Hindsights, MAPS 92 Special. 254p.

[14] Schwarz, N (1987): Einführung in TEX, Addison-Wesley.

[15] Tutelaers, P (1992): A font and a style for typesetting chess using LATEX or TEX. *TUGboat* 13, no. (1), 85–90.

[16] Wirth, N (1976): Algorithms + Data Structures = Programs. Prentice-Hall.

Typesetting Crosswords via TEX, revisited

Kees van der Laan

Hunzeweg 57,
9893 PB Garnwerd, The Netherlands
cgl@rug.nl

October 1992

Abstract

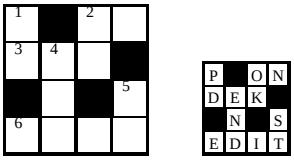
An alternative macro, to van der Laan (1992b), is provided for typesetting crosswords via TEX.

Keywords: Crosswords, games, plain TEX.

1 Introduction

The previous version has been published in the proceedings of EuroTEX '92, and in MAPS92.1. A TUGboat referee suggested not to use `\halign`, et voilà. The differences of this version with the previous version are: `\halign` is not used, and therefore there is no mark-up phase, and `\data` is hidden. The nested FIFO paradigm is directly applied to `\data`, van der Laan (1992a). The control sequence `\data` is created in `\store` with as replacement text the data provided between `\bdata...\edata`. Alternative sizes of the crossword can be obtained via appropriate (re)definition of `\usersize`.

2 Example of use



is obtained via[1]

```
\input crwrev.tex
\bdata%
P*On
DEk*
*n*S
Edit
\edata$$\crw\qquad
\def\usersize{\fiverm\csize=3ex}\sol$$
```

Conventions for `\bdata` ⟨ *data* ⟩ `\edata`[2]

- cell descriptions have to be given line by line
- * denotes crossed out cell
- capitals denote marked open cells (with reference numbers to the clues), and letters of the solution
- lower case letters, denote empty cells, and letters of the solution.

The explicit number of cells is not needed to specify, nor used.

3 Programming

Crossword diagrams consist of (marked) empty cells, crossed out cells, and for the solution cells with (capital) letters. I parameterized each cell into the size `\csize` by `\csize`, with height `.8\csize`. Cells are typeset per row in a `\hbox` and these boxes are stacked in a `\vbox`. All is framed via `\boxit`. The carriage return, `^^M`, and space are active characters between `\bdata` and `\edata`, allowing WYSIWYG input. The numbering of the marked cells is done automatically, row-wise and hidden. I adopted the convention to use reversed words for end-parameter separators, except for the `\edata` separator.

Furthermore, I assumed that no diacritical marks are used in crosswords, and restricted myself to the roman alphabet.

3.1 The file `crwrev.tex`

```
\let\ea=\expandafter \newif\ifpuzzle
\newcount\cnt \puzzletrue
\newdimen\csize\csize=3ex
%
\def\bdata{\bgroup\obeylines\obeyspaces%
\store}
\def\store#1\edata{\egroup\def\data{#1}}
{\obeyspaces\global\let =\relax}
\def\usersize{}
%
```

[1] The sides seem to wiggle. Is this optical illusion, a driver bug, or caused by the properties of ink-blocks on paper? Electronic previewing did not suffer from this. The framing was added via `\boxit` to enhance a straight frame.

[2] A 'white lie,' spaces are also allowed for crossed out cells, see the Appendix.

```
{\catcode‘\^^M=13 %local scope
\gdef\crw{\cnt=0\relax\boxit{\usersize%
\hrule\ea\fifol\data\lofif^^M}}
\gdef\sol{\boxit{\def\num{}\puzzlefalse%
\usersize\hrule\ea\fifol\data\lofif^^M}}
\gdef\fifol#1^^M{\ifx\lofif#1\lofif\fi%
\processl{#1}\fifol}}%end local scope
%
\def\lofif#1\fifol{\fi}
\def\processl#1{\hbox{\fifo#1\ofif}\hrule}
\def\fifo#1{\ifx\ofif#1\ofif\fi%
\process#1\fifo}  \def\ofif#1\fifo{\fi}
%
\def\process#1{\if*#1\cc\else%
\ifx\relax#1\cc\else%
\ifnum‘#1=\uccode‘#1\cap#1\else%
\low#1\fi\fi\fi}
\def\low#1{\hbox to\csize{\vrule
height.8\csize depth.2\csize\relax%
\ifpuzzle\null\else\hss\uppercase{#1}%
\fi\hss\vrule}}
\def\cap#1{\hbox to\csize{\vrule
height.8\csize depth.2\csize\relax%
\num\ifpuzzle\null\else\hss#1\fi%
\hss\vrule}}
\def\cc{\vrule height.8\csize depth%
.2\csize width\csize}
%
\def\num{\global\advance\cnt1\relax%
\vbox to.8\csize{\rlap{\kern1pt%
\fiverm\the\cnt\hss}\vfil}}
%
\def\boxit#1{\vbox{\hrule\hbox{\vrule%
\vbox{#1}\vrule}\hrule}}%   cgl, oct92
```

3.2 \crw, \sol

The crossword, respectively the solution are typeset (as \vbox-es) by these macros. \data is used.

3.3 \process

This macro typesets each cell contents according to the \data. \if*#1 etc. tests whether a crossed-out cell has to be typeset, and if so a copy is inserted. For the other situation according to the case of the letter \low(er case letter) or \cap(ital letter), is invoked. How the cell contents will be typeset depends upon \ifpuzzle. The letters are typeset in upper case and centered.

3.4 \num

Generates and typesets the reference numbers. The numbers are set in the left upper corners of the cells marked by capitals in the puzzle representation.

3.5 \bdata, \store

These store the user provided information between \bdata and \edata in \data, with the carriage return and space as active characters. (The \bgroup after \gdef must be an *explicit* brace, because \bgroup can be equally well a parameter separator.)

References

[1] Hamilton Kelley, B (1990): Some macros to draw crosswords. *TUGboat* 11, no. (1), 103–119.

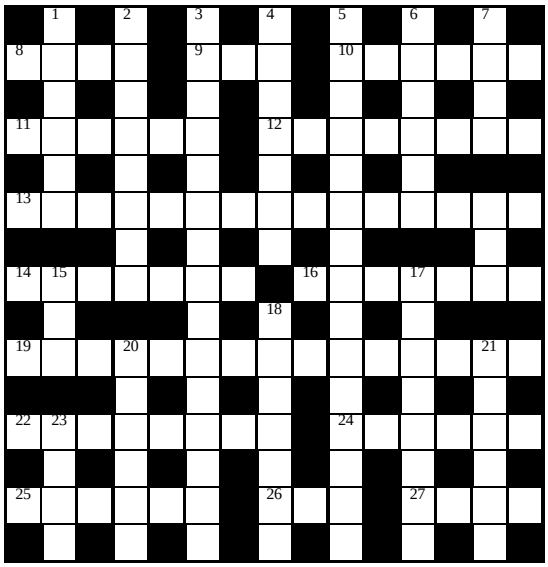
[2] Knuth, D.E (1986): The TEXBook. Addison-Wesley.

[3] Laan, C.G van der (1992a): FIFO and LIFO sing the BLUes. MAPS92.2.

[4] Laan, C.G van der Laan (1992b): Typesetting Crosswords via TEX. EuroTEX ’92, 217–224. Also in MAPS92.1.

Appendix

Hamilton Kelley’s puzzle



is obtained—after \input crwrev.tex—via

```
\bdata%BHK’s example
 S I C T D S P*
Swam Oho Icecap
 o p m r t n l*
Bopeep Schedule
 s l a i y u  *
Thalassographer
   e s n a   r*
HAirpin UmbRage
 r   o S b i  *
ScaLenetriangLe
   o a u c g e*
AMounted Allege
 a v e e l e a*
Floral Nil Tace
 l e y t y s y*
\edata$$\crw$$
```

Scientific Word; TEX à la WYSIWYG

Dr. Jan Krugers*

Gagelweg 3
4651 VL Steenbergen

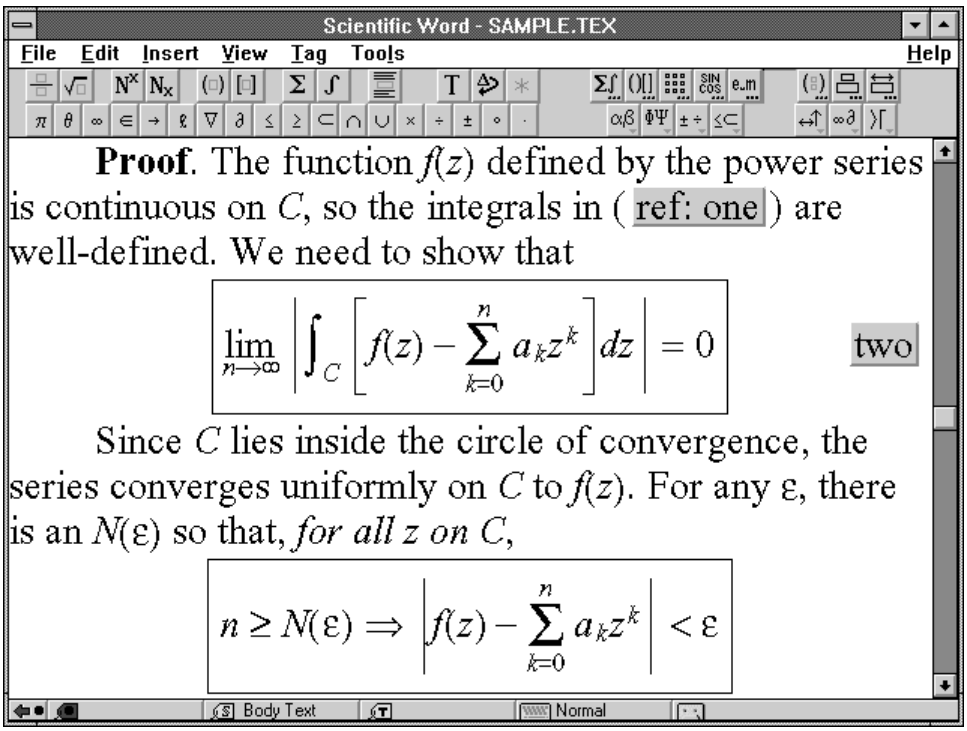
Abstract

A software package for Windows and OS2WIN for inputting mathematical formulas WYSIWYG according to all TEX rules. Internal storage format is a mixture of TEX, LATEX and macros of own design. Mathematical formulas are immediately shown the way they will be printed. Line width for text is limited to the width of the window for ease of use. The built in previewer from TurboTEX shows the whole document layout. There are also dvi drivers included for PostScript, LaserJet, DeskJet and matrixprinters. Pictures can be called just as easy as formulas can be inputted. A description of the philosophy of this pacakge and how to work with it.

1 Introduction

SCIENTIFIC WORD is a Windows front-end to LATEX. It is a full document editor, not just an equation editor. The SCIENTIFIC WORD package includes a complete implementation of TEX for Windows. If you want the outstanding typeset output provided by TEX, but don't want to use the TEX language, SCIENTIFIC WORD is the word processor for you.

In SCIENTIFIC WORD, you interact directly with mathematics in familiar form, not with TEX codes. You can create your document at the computer with no intermediate coding step. Making changes and corrections is equally straightforward.



Scientific Word screen at 2x view

*In cooperation with TCI Software Research

1.1 Save Time and Increase Your Productivity

SCIENTIFIC WORD will save you time in five essential ways:

1. Time spent entering and correcting TEX codes is "dead time" during which you cannot think about the content of the document itself. By eliminating the coding and by showing mathematics on-screen in familiar form, SCIENTIFIC WORD let's you compose your document directly at the computer.
2. SCIENTIFIC WORD eliminates the need to print or preview a document to check that it is correctly coded. A Windows previewer is provided for you to check the final printed appearance.
3. Some word processors include an equation editor. While they are fine to type an occasional equation, you waste time going back and forth between the body text and equation frames. With SCIENTIFIC WORD, you enter text and mathematics on one continuous screen so you don't waste time popping in and out of equation boxes.
4. By handling all of the spacing, by automatically italicizing mathematics, and by providing intelligent objects like fractions, radicals and matrices, SCIENTIFIC WORD saves you the tedious job of hand formatting which often leads to inconsistent results.
5. Unlike other Windows word processors which focus on the appearance of text, SCIENTIFIC WORD focuses on the meaning or content. If you've ever been through the process of changing the format of an even moderately sized document with a WYSIWYG (What You See Is What You Get) system, you'll appreciate that SCIENTIFIC WORD's revolutionary approach has the potential to save you an extraordinary amount of time. The small initial investment in "unlearning" the habits developed for WYSIWYG systems is handsomely rewarded in a very short time. For more details, see the section entitled **Logical Design**.

1.2 Objects

SCIENTIFIC WORD is object-oriented. That means the program deals with familiar objects you understand, such as fractions, matrices, brackets, and radicals. You'll find these objects have specific properties and that they behave intelligently.

For example, the numerator and denominator of a fraction are automatically centered and the fraction bar expands as you type. Brackets and radicals also expand or contract as you enter an expression. Because the objects themselves are intelligent and adjust themselves as you build a complex expression, it makes your job of entering mathematics that much easier.

Furthermore, SCIENTIFIC WORD is an intelligent interpreter of what you type. If you are in mathematics and type cos, the software recognizes it as a function and stores and displays it accordingly. And SCIENTIFIC WORD selects the correct typeface, spacing and placement of limits required by functions as you type.

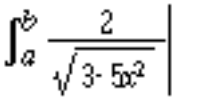
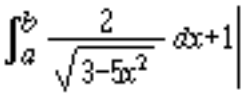
2 Entering Mathematics—An Example

In the following two tables, we describe two ways you can enter the expression $\int_a^b \frac{2}{\sqrt{3-5x^2}}\,dx + 1$ in SCIENTIFIC WORD. The first uses the mouse wherever possible and the second uses the keyboard only. In this example, you will see that you add limits to an operator like $\int$ by applying a subscript and a superscript. An alternative approach taken by other systems is to provide a special icon with the limits already present. This leads to a bewildering array of icons on the screen. The SCIENTIFIC WORD approach uses a much smaller set of icons. Another large reduction in the number of icons comes from the unique feature that the size and position of limits on operators like $\sum$ change automatically from their in-line form $\sum_{n=1}^{\infty}$ to their display form

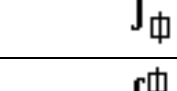
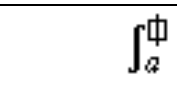
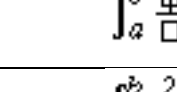
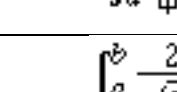
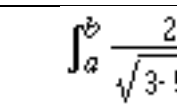
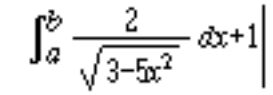
$$\sum_{n=1}^{\infty}.$$

Here are the steps to enter $\int_a^b \frac{2}{\sqrt{3-5x^2}}\,dx + 1$ using the mouse.

Entering an Integral Using the Mouse	
Action	**Result**
Click 	
Click 	

Action	Result
Type a, click to the right, click 	
Type b, click to the right, click 	
Type 2 and click in the denominator box	
Click 	
Type $3-5x$, click  type 2, click to the right	
Type $dx+1$	

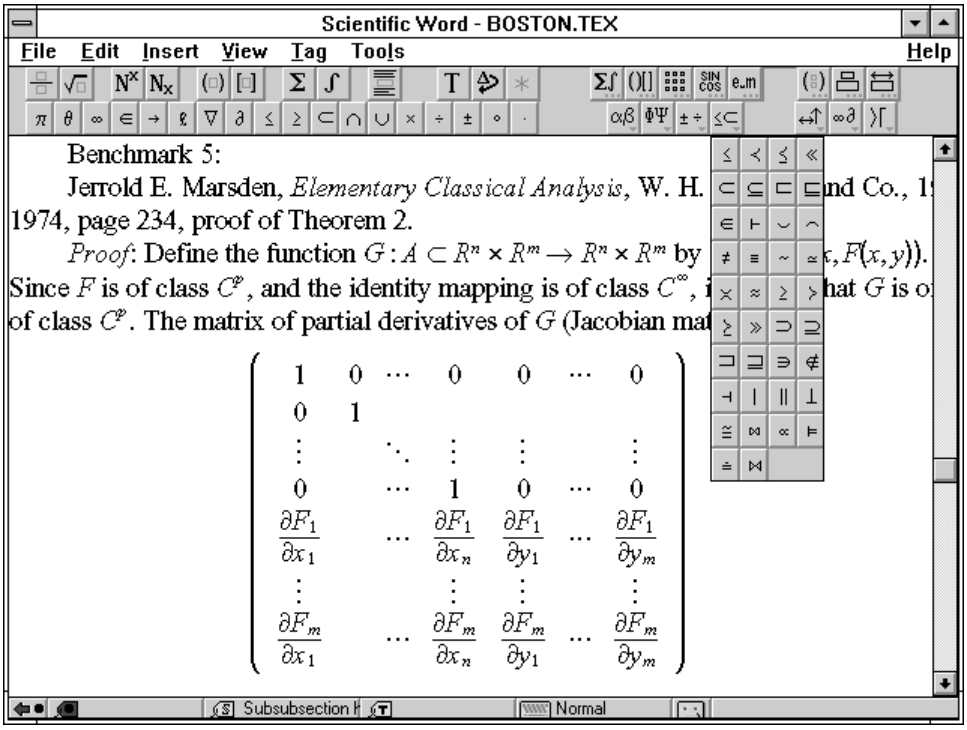
Here are the steps to enter the same formula $\int_a^b \frac{2}{\sqrt{3-5x^2}}\,dx+1$ directly from the keyboard.

Entering an Integral Using the Keyboard	
Action	**Result**
Press *ctrl+i*	
Press *ctrl+l* or *ctrl+down arrow*	
Type a, press *tab*	
Type b, press *spacebar*, press *ctrl+f*	
Type 2, press *down arrow*	
Type *ctrl+r*	
Type $3-5x$, press *ctrl+H*, type 2, press *end*	
Type $dx+1$	

The number of keystrokes is small and follows the same logic as the mouse operations. SCIENTIFIC WORD is designed so that you can get started using the mouse with very little effort. Once you learn the keyboard shortcuts, you can enter text and mathematics very quickly without the mouse.

3 Available Symbols

The available symbols are grouped in seven drop-down panels. The symbol panels drop down from the icon area at the top of the screen:



The binary relations symbol panel

The panels and their contents are as follows:

3.1 Lowercase Greek

$\alpha\beta\gamma\delta\epsilon\zeta\eta\theta\vartheta\iota\kappa\lambda\mu\nu\xi\pi\varpi\rho\sigma\varsigma\tau\upsilon\phi\varphi\chi\psi\omega$

3.2 Uppercase Greek

$\Gamma\Delta\Theta\Lambda\Xi\Pi\Sigma\Upsilon\Phi\Psi\Omega$

3.3 Binary Operations

$\pm\ \mp\ \setminus\ \cdot\ \times\ *\ \star\ \diamond\ \circ\ \bullet\ \div\ \triangleleft\ \vee\ \wedge\ \oplus\ \ominus\ \otimes\ \oslash\ \cap\ \cup\ \uplus\ \sqcap\ \sqcup\ \triangleleft\ \triangleright\ \wr$
$\bigcirc\ \triangle\ \triangledown\ \triangleright\ \odot\ \dagger\ \ddagger\ \amalg\ \trianglelefteq\ \trianglerighteq$

3.4 Binary Relations

$\leq\ \prec\ \preceq\ \ll\ \subset\ \subseteq\ \sqsubset\ \sqsubseteq\ \in\ \vdash\ \smile\ \frown\ \neq\ \equiv\ \sim\ \simeq\ \asymp\ \approx\ \geq\ \succ\ \succeq\ \gg$
$\supset\ \supseteq\ \sqsupset\ \sqsupseteq\ \ni\ \notin\ \dashv\ |\ \|\ \perp\ \cong\ \bowtie\ \propto\ \models\ \doteq\ \Join$

3.5 Arrows

$\leftarrow\ \Leftarrow\ \rightarrow\ \Rightarrow\ \leftrightarrow\ \Leftrightarrow\ \mapsto\ \hookleftarrow\ \leftharpoonup\ \leftharpoondown\ \rightleftharpoons\ \uparrow\ \Uparrow\ \downarrow\ \Downarrow\ \updownarrow\ \longleftarrow\ \Longleftarrow\ \longrightarrow\ \Longrightarrow$
$\longleftrightarrow\ \Longleftrightarrow\ \longmapsto\ \hookrightarrow\ \rightharpoonup\ \rightharpoondown\ \rightleftharpoons\ \leadsto\ \Updownarrow\ \nearrow\ \searrow\ \swarrow\ \nwarrow$

3.6 Miscellaneous

$\ldots\ \cdots\ \vdots\ \ddots\ \int\ \aleph\ \hbar\ \imath\ \jmath\ \ell\ \wp\ \Re\ \Im\ \partial\ \infty\ \Box\ \forall\ \exists\ \neg\ \flat\ \natural\ \sharp\ \emptyset\ \nabla\ \surd\ \top\ \bot\ \|\ \angle\ \triangle\ \diamond$
$\sharp\ \clubsuit\ \diamondsuit\ \heartsuit\ \spadesuit$

3.7 Special Delimiters

$\lfloor\ \rfloor\ \lceil\ \rceil\ \langle\ \rangle$

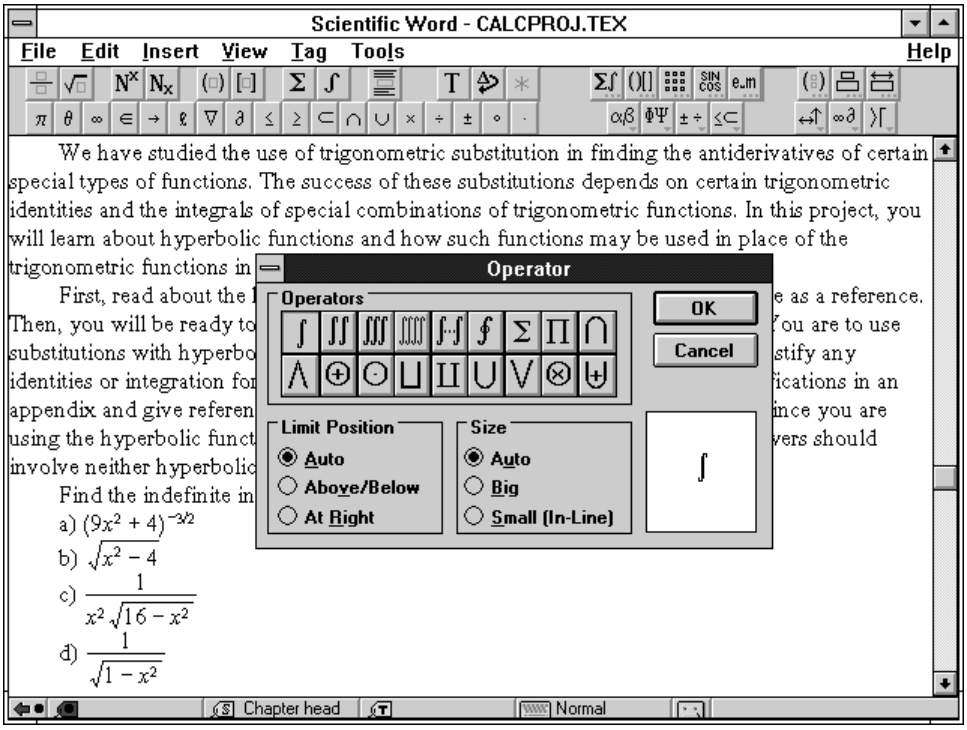
Symbols can also be typed directly from the supplied Greek and symbol keyboards. Alternatively, if you know the TEX name of a symbol, you can type it directly. For example, to enter α, you hold down the *ctrl* key and with the key held down, type `alpha`.

4 Text and Mathematics

SCIENTIFIC WORD treats plain text in much the same way as any other word processor, following the standard Windows conventions for inserting, deleting, selecting, cutting, copying and pasting. To switch from text to mathematics, you click the Math/Text icon at the top of the screen, or press *ctrl+m*. While you are in mathematics, characters are italicized, functions like sin and cos are recognized automatically, and the spacing is supplied around binary operations and relations like + and =. You can convert any selection to or from mathematics by clicking the Math/Text icon. By default, mathematics is shown in red on the screen. This feature makes it easy to discern and read the mathematics, and also makes clear whether punctuation is or is not part of the mathematics.

5 More Mathematics

The most frequently used mathematical objects are available with a single mouse click. A complete range of objects is available through SCIENTIFIC WORD's dialogs. Here is the operator dialog which gives you the full set of large operators available in $\mathcal{AMS}$-TEX.



The Operator dialog

To create a mathematical display, you click the display icon and type the mathematics. Or, you can select the mathematics and click the display icon. A display can be set up so that it is automatically numbered. Displays can be given symbol names so that they can be cross-referenced in the body of the document. When the document is printed, the cross-reference is replaced by the automatically generated equation number.

Complex expressions are easy to create. The process is made even easier by the ability to apply radicals, brackets, fractions, and displays to selections. For example, if you have already typed a large expression and you want to apply brackets to all or part of it, you select that part, then click on the brackets icon.

You enter a matrix by specifying the number of rows and columns. Typing the entries is the same as entering any other mathematics in SCIENTIFIC WORD. You can select rows or columns and delete them. You can left, center, or right align columns, and you can align the entire matrix vertically at its center, at the top row, or at the bottom row. It is a simple matter to insert additional rows or columns in an existing matrix.

The document you are now reading was created with SCIENTIFIC WORD. The following sample expressions were created using the operations discussed above.

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 1 & & & & & \\ \vdots & & \ddots & \vdots & \vdots & & \vdots \\ 0 & & \cdots & 1 & 0 & \cdots & 0 \\ \dfrac{\partial F_1}{\partial x_1} & & \cdots & \dfrac{\partial F_1}{\partial x_n} & \dfrac{\partial F_1}{\partial y_1} & \cdots & \dfrac{\partial F_1}{\partial y_m} \\ \vdots & & & \vdots & \vdots & & \vdots \\ \dfrac{\partial F_m}{\partial x_1} & & \cdots & \dfrac{\partial F_m}{\partial x_n} & \dfrac{\partial F_m}{\partial y_1} & \cdots & \dfrac{\partial F_m}{\partial y_m} \end{pmatrix}$$

$$iB_r\left[\tilde{a}_{kl}^n\right] \equiv \frac{1}{2}(h_{n-1}+h_n)\int_0^R 2\pi r\,dr\left[\rho_k^{n*}(r)\right]\frac{d}{dr}\left[\Psi_l^n(r)\right],$$

$$\left[D_{r,kl}^n\right]^{-1} \equiv \int_0^R 2\pi r\,dr\int_{z_n-\frac{1}{2}h_{n-1}}^{z_n+\frac{1}{2}h_n}\left[\rho_k^{n^*}(r)\right]\left[D^{-1}(r,z)\right]\left[\rho_l^n(r)\right],$$

$$\psi = -\frac{\kappa\varpi\varpi'}{4\pi}\int_0^{2\pi}\frac{\cos(\theta-\theta')d\theta}{\sqrt{\{(x-x')^2+\varpi^2+\varpi'^2-2\varpi\varpi'\cos(\theta-\theta')\}}}$$

$$1/f(x) = \frac{1}{c_0}\left[\frac{1}{1+\left(\sum_1^\infty+\sum_{-\infty}^{-1}\right)\frac{c_n}{c_0}e^{inx}}\right]$$

$$\prod_{j\geq 0}\left(\sum_{k\geq 0}a_{jk}z^k\right) = \sum_{n\geq 0}z^n\left(\sum_{\substack{k_0,k_1,\ldots\geq 0\\ k_0+k_1+\cdots=n}}a_{0k_0}a_{1k_1}\cdots\right)$$

$$\left\{\left(\left[\left\|\sum\right\|\right]\right)\right\}$$

$$\hat{a}\,\hat{A}\check{a}\,\check{A}\tilde{a}\,\tilde{A}\acute{a}\,\acute{A}\grave{a}\,\grave{A}\dot{a}\,\dot{A}\ddot{a}\,\ddot{A}\breve{a}\,\breve{A}\bar{a}\,\bar{A}\vec{a}\,\vec{A}$$

$$\tau \stackrel{\text{def}}{=} \frac{\mathbf{R}\cdot(\ddot{\mathbf{R}}\times\dddot{\mathbf{R}})}{\left\|\mathbf{R}\times\ddot{\mathbf{R}}\right\|^2}$$

$$\min_{u\in U}\max_{v\in V}\Phi(u,v)\geq\max_{v\in V}\min_{u\in U}\Phi(u,v)$$

$$\hat{\beta}=\frac{\sum_{i=1}^{m}x_iy_i-n\bar{x}\bar{y}}{\sum_{i=1}^{n}x_i^2-n\bar{x}^2}=\frac{\sum_{i=1}^{m}x_iy_i-\frac{1}{n}\left(\sum_{i=1}^{m}x_i\right)\left(\sum_{i=1}^{m}y_i\right)}{\sum_{i=1}^{m}x_i^2-\frac{1}{n}\left(\sum_{i=1}^{m}x_i\right)}$$

6 Fragments

You can select a portion of a document (called a fragment) and save it for later recall. This macro-like feature is useful for creating a library of frequently-used formulas and phrases. To recall a fragment, you place the cursor where you want to insert the fragment, then click on the fragment icon at the bottom of the screen. This pops up a list of fragments to choose from.

7 Document Structuring

SCIENTIFIC WORD provides a direct interface to the document structuring features of LATEX. To create a section heading, you type the text of the heading, select it, then click on Section in the list of sectioning tags which pop up from the bottom of the screen.

Items in lists, theorem statements, and bibliography entries are all created in the same way—by applying the appropriate tag to a selection. Cross-references and citations are entered as special fields.

The components of the document front matter, including the title, abstract, and list of authors, are entered into a separate section of the document. A table of contents is generated automatically by checking an entry in the dialog box which configures the document front matter.

8 TEX, LATEX and Scientific Word

As those of you familiar with TEX already know, TEX is the mathematical typesetting system created by Donald Knuth and supported by the American Mathematical Society. It is the acknowledged leader in high-quality scientific typesetting. LATEX is a TEX macro package created by Leslie Lamport. LATEX enhances TEX with document structuring features such as lists, automatic table of contents generation, cross-references, automatic theorem numbering, bibliographies, and tables. The Windows implementation of TEX and LATEX supplied with SCIENTIFIC WORD is called TurboTEX, a product of Kinch Computer Company.

By using TEX for printing, SCIENTIFIC WORD takes advantage of the many automatic typesetting features available within TEX, such as kerning, hyphenation, and automatic sizing, placement, and positioning of mathematics.

Some features of TEX and LATEX are not directly accessible in the first release of SCIENTIFIC WORD but will be added in future releases. These features include tables, footnotes, and marginal notes. To handle these cases in the first release, SCIENTIFIC WORD provides a TEX field for direct entry of LATEX commands. Using TEX fields requires a knowledge of TEX and LATEX. A selection of pre-defined TEX fields for the most common items is supplied in the form of recorded fragments.

Future releases will include a direct interface for entering tables, footnotes and marginal notes so that the user will not be required to use TEX fields to take advantage of these features.

9 Logical Design

The heart of the SCIENTIFIC WORD approach is the separation between content and appearance. The purpose of writing is to convey ideas, so the creative process of forming ideas and putting them into words should be the writer's primary concern. The mechanical process of displaying the document on the printed page in the most readable and organized format is a function of typesetting, not writing, and should not be the writer's primary concern.

SCIENTIFIC WORD's separation of the creative process of writing from the mechanical process of layout is known as *logical design*[1]. Logical design results in greatly increased productivity because the writer is freed from formatting concerns and can focus on creating content. Logical design also leads to better, more consistent typeset results because choices of fonts, spacing, emphasis, and countless other aspects of the document's appearance are made automatically. Before SCIENTIFIC WORD, this approach was only available in batch systems like TEX, LATEX, and production typesetting systems which use markup languages. The unique feature of SCIENTIFIC WORD is that it provides a graphical interface to logical design.

An alternative approach to word processing, called *visual design*, is to make the screen look as close to the printed page as possible. This approach has also been called WYSIWYG for What You See Is What You Get. It is the approach taken by T^3, Ami Pro, Word for Windows and WordPerfect for Windows.

Why is it important that you understand the difference between logical design and visual design? Because a logical system requires a very different interface from a visual system. If you are familiar with a WYSIWYG

[1] This concept of "logical design" vs "visual design" is used by Leslie Lamport in his book, the LATEX User's Guide and Reference Manual.

system, you may at first be surprised by some of the differences:

- WYSIWYG systems divide documents into pages according to their anticipated typeset appearance. To see an entire line, you often have to scroll horizontally because the screen dimensions and page dimensions do not match. In a logical design, working with pages is unnecessary, since the division of a document into pages has no connection with the document's logical structure. Thus, SCIENTIFIC WORD breaks lines to fit the window. If you resize the window, SCIENTIFIC WORD reshapes the text to fit. You never need to scroll horizontally.
- In WYSIWYG systems, you are constantly giving commands which affect the appearance of text—you select text and then select a font, point size, or typeface. You apply alignment commands like center, left justify, and right justify. To center an equation, you select it and choose the center alignment. In SCIENTIFIC WORD, all of these commands are replaced by commands which define the logical structure instead of the appearance. There is no command to center—it is replaced by commands to create a title, to create a section head, and to create a displayed equation. The format of the title, the alignment of section heads, and the alignment of equations, are all determined separately by the print style, which can be changed in an instant. SCIENTIFIC WORD comes with a range of pre-defined styles. If none of these fits your needs, TCI will endeavour to create one for you. LATEX style experts can create their own and add them to the list of predefined styles available within SCIENTIFIC WORD.
- In systems that focus on appearance instead of content, pressing the *spacebar* inserts a space that shows on the screen and is duplicated in print. Pressing the *spacebar* a second time inserts a second space. In a logical system like SCIENTIFIC WORD, a space is the separation between words. Pressing the *spacebar* a second time has no meaning, so it has no effect.

SCIENTIFIC WORD's emphasis on logical structure doesn't ignore the fact that documents must still be printed in a readable, organized, and visually pleasing format. SCIENTIFIC WORD provides a variety of article, report, and books styles that you can apply to your document to produce a high-quality, typeset appearance in print. Because you must be able to see how a document will look without actually printing it, SCIENTIFIC WORD provides an on-screen preview feature.

10 Scientific Word File Formats

SCIENTIFIC WORD stores documents as ASCII files in LATEX format. Wherever possible, SCIENTIFIC WORD writes standard LATEX commands. In cases where this is not possible, SCIENTIFIC WORD uses special macros supplied with the system. These include macros for mathematical constructs not directly supported in LATEX, and macros for importing graphics.

Most existing LATEX documents can be read into SCIENTIFIC WORD.

SCIENTIFIC WORD imports documents created by T^3's T3toTeX converter.

11 Scientific Word's Mathematics

The mathematical objects in SCIENTIFIC WORD are designed to provide the simplest and most natural editing possible. When we read plain text, we deal with whole words at a time. However, the best and most natural text editors deal with individual characters at the lowest level—the cursor moves between the letters of a word and you make changes at the lowest level by inserting or deleting individual characters. Although there are hundreds of thousands of words in the English language, there are only 26 characters and a few punctuation symbols which are the "atoms" of plain text editing. In the same way, SCIENTIFIC WORD provides a small set of mathematical "characters" as the atoms for the two-dimensional structure of familiar mathematical notation. This notation is used to represent thousands of mathematical objects and concepts.

For example, the SCIENTIFIC WORD formula $\int_a^b \frac{2}{\sqrt{3-5x^2}}\,dx + 1$ is a five-element list: an integral operator, a fraction, the word dx, the binary operator $+$, and the word 1. The integral operator is a template containing the two one-element lists a and b. The fraction is a template containing the one-element list 2 and a one-element list containing a radical. The radical is a template containing a four-element list: the word 3, the binary operator $-$, the word $5x$, and a superscript. The superscript contains the one-element list 2. The atoms are the operator, fraction, radical, and superscript. The cursor moves naturally through this structure, and you can edit it without constraint.

This approach avoids the awkwardness of systems which insist, for example, that operators have a distinct argument component ($\frac{2}{\sqrt{3-5x^2}}$ in our example), forcing the user to be aware of invisible boundaries. To get around this, the argument region is surrounded by a dotted box on screen, cluttering the screen and making the mathematics unreadable.

Another feature of SCIENTIFIC WORD's objects is that the empty input boxes of templates like operators are suppressed as much as possible, again with the goal of making the mathematics look as natural as possible at all times to enhance readability.

12 Summary

Scientific Word is a Windows based software package that makes it possible that even the non-knowledgeable TEX-person can write TEX documents! The knowledgeable TEX-person can add to the internal storage format his knowledge, either by making small changes or by adding macros or complete style sheets to the many

provided. The price of the package is H.Fl. 2000,–.[2] Further information[3] can be obtained from Technical Marketing Consulting, Gagelweg 3, 4651 VL Steenbergen, phone 01670-64422 and fax 01670-66555.

A Note added in proof by Kees van der Laan

TEX à la WYSIWYG does appeal. Like similar systems it suffers from the following general drawbacks.

- **Cost-effectiveness**
  It is doubtful whether the keyboarding is the most expensive part of (La)TEX-ing, see experience $\mathcal{AMS}$. Syntactical correction and fine-tuning—that is adaptation of the results to the traditional high-quality proofs—is hard.
- **Incompleteness**
  What about graphics? What about tables? What about programmable items? What about general constructs, for example figures, which are reused, slightly modified? And what about real difficult things like commutative diagrams?
- **Entering vs. correction**
  What about correcting/adapting an already existing document?
- **Context dependence**
  From the example on the first page the integral sign is too small and does not 'know of' the size of the integrand. The systems I have seen lack context awareness.
- **Interfacing with other tools**
  What about inclusion of encapsulated PostScript? What about Metafont prepared graphics à la 'When Metafont and TEX work together?'
- **High-Quality?**
  For high-quality typesetting math explicit kernings and awareness of formula classes are needed . The classical example is ':' as operator or as separator.
  Furthermore, the general advice to keep the typesetting as simple as possible is neglected. Confer the example of $1/f(x)$, and the representation of $\hat{\beta}$.
- **Formats**
  What about the role of formats c.q. sty-fyles?
- **Portability**
  Can resulting documents easily be exchanged via e-mail?

Remember that this kind of software needs a powerful machine, while elementary processing —$plain+know-how$— can be done on a very modest equipped machine, and is portable.

B Note added by MAPS editor

Up to now, I have the following experience with this product:

- The paper in this issue of the MAPS was originally generated by SCIENTIFIC WORD. Both LATEX files, style files and encapsulated PostScript files were send by author to the editor using diskettes. After a minor correction within the SCIENTIFIC WORD style file, the pictures were included in the MAPS without any problem.
- The main function of SCIENTIFIC WORD is in my opinion the easy generation of mathematics LATEX code, especially for the novice and occasional user. Besides that, the software could be a first introduction for the use of LATEX, or can be used by the non-LATEX interested (or WordPerfect minded) user community.
- This kind of tools could force the user not to start directly with TEX. As most of my colleagues knows, your MAPS editor strongly discourage the use of *plain* TEX, both by novice and experienced users. The only function of plain TEX should be the construction of building blocks (objects/macros) in order to use these in a logical (con)text environment (LATEX).

In additions to this, A. Al Dhahir forwarded me the following information:

- SCIENTIFIC WORD is based on LATEX, with corresponding restrictions.
- It is not really WYSIWYG. On the screen you see PostScript ATM fonts (also for mathematics). In the printing or preview mode, CM fonts are used.
  SCIENTIFIC WORD has an interface with TurboTEX for windows, and works as a layer between Windows and LATEX of TurboTEX.
- Tables were not used, but they can be expected in a next release.

C Note added by author

- **Cost-effective for the average user**
  SCIENTIFIC WORD is intended to make it easy for the user to input text and formula's. The resulting document can be fine-tuned within SCIENTIFIC WORD to a large degree. But for the final publication the publisher with its elaborate practical experience can and will fine-tune the document created by SCIENTIFIC WORD anyway.
- **Incompleteness**
  Graphics are supported and tables will be fully supported in the next release. General constructs can be stored as keyboard macro's and be recalled.
- Entering and correction
  A TEX document under SCIENTIFIC WORD is much easier to correct than a document without a WYSIWYG interface. In many instances an existing TEX document can be read into SCIENTIFIC WORD for correction.
- **Context dependence**
  SCIENTIFIC WORD has full context awareness. Size of subscript and superscript is dependent on the contents like for in-line or display formula's. Due to the limited screen resolution that may not always be visible on the screen but it is there on the print out.
- **Interfacing with other tools**
  SCIENTIFIC WORD does import Encapsulated PostScript files not Metafont files.
- **High Quality**
  High quality can be accomplished but the average user can and will leave this to the final publisher, see under Cost-effective.
- **Formats**
  SCIENTIFIC WORD is delivered with many style files. With TEX knowledge they can be changed.
- **Portability**
  There is freely available a special utility to 'pack' a SCIENTIFIC WORD file into a format suited for e-mail. Obviously it can do the reverse too.

[2] Special for NTG members there will be a 25% discount if they order the software before the end of this year. More details about this offer can be found elsewhere in this MAPS.

[3] In the next issue of the MAPS, an evaluation of Scientific Word by one of the NTG members, will be published

Bugs (sigh) in Knuths 'Computers & Typesetting'

Gerard van Nes (editor)

October 1992

1 Introduction

In may/june this year a discussion started on the TEX-NL listserver (by Kees van der Laan and Nico Poppelier) about the releases of Knuths book series: it was not sure that Addison-Wesley was selling only the latest book editions.

The problem is that references to the corresponding software release (the software changes strongly in TEX 3.x and METAFONT 2.7) are missing at the cover of the books. Besides that it is not clear which 'edition' and/or 'number' and/or 'ISBN' reflects the latest TEX (3.x) or METAFONT (2.7) software. The only way to check the actual version is e.g.:

- Does the index contains `\language`?
- Does the index contains `\emergencystretch`?

We have forwarded that discussion directly to Addison-Wesley in Amsterdam. The answer is included in next section.
Rita Snaddon informed me also that for 'The TEX Book' (Volume A; pb), the 22st printing will be expected very soon. This will be the final printing corresponding with the current book number.

The upgrades of the books since 1984 are described in a set of errata sheets (section 4). These errata sheets can directly be ordered by Addison Wessley (attn Rita Snaddon) but are also available by ftp (see section 3).

2 Answer from Addison-Wesley

Rita Snaddon from Addison-Wesley[1] informed me about the printing dates of the books with the following message:

After much research into the problem of the Knuth 'Computers and Typesetting' Series reprints, I hope that I have now come up with a solution to your satisfaction.

If I can explain: Normally when a book is reprinted with major changes, it turns into a new edition with a new ISBN number. The old editions can then normally be destroyed. When a book is reprinted with major changes (in case of the TEX and METAFONT books), but does not go into a new edition, we still keep stock of the older prints. It's an unusual problem which we do not encounter often.

I have obtain Errata Sheets from Donald Knuth that detail all the changes in the following volumes. If you would like to notify your NTG members about these erratas, we will be pleased to send them out free of charge.

Below is an overview of the latest printings:

Code	Vol	Title	Edition	Number	Date
13447	A	The TEX Book	1	12th	3/92
13437	B	TEX:The Program	1	4th	5/91
13445	C	The METAFONT Book	1	4th	9/91
13438	D	METAFONT: The Prog	1	4th	2/92
13446	E	Comp. Modern Type	1	3rd	9/92
13448	A	The TEX Book (pb)	1	21st	5/92
13444	C	The METAFONT Book (pb)	1	7th	5/92

[1] Rita Snaddon, Sales Support Manager, Addison-Wesley, Concertgebouwplein 25, 1071 LM Amsterdam, phone: 020-6717296, fax: 020-6645334

3 Availability of errata sheets

Errata steets for Knuths 'Computers & Typesetting' are available by anonymous ftp at:

`labrea.stanford.edu`

Directory:

`pub/tex/errata`

This directory contains the following files:

```
 24569 Aug 11  1989 errata.one
 15006 Aug 11  1989 errata.two
 60085 Aug 11  1989 errata.three
 31881 Sep 26  1989 errata.four
 42967 Mar 16  1992 errata.five
 15838 Jan 29  1991 errata.six
 29683 Mar 16  1992 errata.seven
  2473 Mar 16  1992 errata.tex

  4110 Oct 19  1991 logmac.tex
 87397 Jan 25  1992 mf84.bug
 56228 Mar  6  1992 cm85.bug
305187 Mar 16  1992 tex82.bug
150372 Mar 16  1992 errorlog.tex
```

4 Contents of ftp files

In the following subsections, the headers of the errata sheets are listed.

errata.one

This is a list of all corrections made to *The TEXbook* between the first and second printings. If your copy says '*Second printing (October 1984)*' on the copyright page, you've already got all of these things corrected. Otherwise, you're a lucky owner of the rare first edition; read on.

errata.two

This is a list of all corrections made to *The TEXbook* since the second printing. If your copy doesn't say '*Second printing (October 1984)*' on the copyright page, you should also look at the previous bug list. In fact, the most important corrections to the first printing were discovered first, so they've already been made.

errata.three

This is a list of all corrections made to *Computers & Typesetting*, Volumes A–E, between the date of publication (May, 1986) and 15 June 1987. It also includes corrections made to the softcover version of *The TEXbook*, beginning with the sixth printing (January 1986); these are the same as corrections to Volume A. Corrections to the softcover version of *The* METAFONT *book* are the same as corrections to Volume C.

errata.four

This is a list of all corrections made to *Computers & Typesetting*, Volumes A–E, between 16 June 1987 and 20 February 1989. Corrections made to the softcover version of *The TEXbook* are the same as corrections to Volume A. Corrections to the softcover version of *The* METAFONT *book* are the same as corrections to Volume C. Some of these corrections have already been made in reprintings of the books. Some of these corrections affect the indexes and mini-indexes of Volumes B and D in ways not shown here. Corrections made up to 15 June 1987 appear in other files.

errata.five

This is a list of all corrections made to *Computers & Typesetting*, Volumes A–E, between 20 February 1989 and 30 September 1989 (when TEX Version 3.0 and METAFONT Version 2.0 were fully defined). Corrections made to the softcover version of *The TEXbook* are the same as corrections to Volume A. Corrections to the softcover version of *The* METAFONT *book* are the same as corrections to Volume C. Some of these corrections have already been made in reprintings of the books. Several minor changes to Volumes A and C are not shown here because they simply make room for the more substantive changes needed to describe the new features of TEX Version 3.0 and METAFONT Version 2.0. Hundreds of changes will soon be made to Volumes B and D because of the upgrades to TEX and METAFONT; it will unfortunately be impossible to document all of those changes. Therefore, readers who need up-to-date information on the TEX and METAFONT programs should refer to the |WEB| source files until new printings of Volumes B and D are issued.

errata.six

This is a list of all corrections made to *Computers & Typesetting*, Volumes A, C, and E, between 30 September 1989 (when the revisions for TEX Version 3.0 and METAFONT Version 2.0 were made) and December 31, 1990. Corrections made to the softcover version of *The TEXbook* are the same as corrections to Volume A. Corrections to the softcover version of *The* METAFONT *book* are the same as corrections to Volume C. Some of the corrections below have already been made in reprintings of the books. Hundreds of changes, too many to list here, have been made to Volumes B and D because of the upgrades to TEX and METAFONT. Readers who need up-to-date information on the TEX and METAFONT programs should refer to the |WEB| source files until new printings of Volumes B and D are issued.

errata.seven

This is a list of all corrections made to *Computers & Typesetting*, Volumes A, B, C, and D, between 1 January 1991 and 15 March 1992. Corrections made to the softcover version of *The TEXbook* are the same as corrections to Volume A. Corrections to the softcover version of *The* METAFONT *book* are the same as corrections to Volume C. Some of the corrections below have already been made in reprintings of the books. Changes to Volume B refer to the fourth printing (1991),

which differs markedly from earlier printings because it includes all the revisions for TEX3.0. Changes to Volume D refer to the third printing (1991), which differs markedly from earlier printings because it includes all the revisions for METAFONT 2.0. Changes to the mini-indexes and master indexes of Volumes B and D are not shown unless they are not obviously derivable from what has been shown. Dozens of changes, too many to list here, have been made to Volume E because of recent upgrades to the Computer Modern font source files. Those changes, which affect only the digitization at low resolution and the appearance of lowercase delta and some characters in the math symbols fonts (but not the `TFM` files), are documented at the end of file `cm85.bug`.

`errata.tex`

This is a list of all corrections made to *Computers & Typesetting* since 15 March 1992. Corrections made to the softcover version of *The TEXbook* are the same as corrections to Volume A. Corrections to the softcover version of *The* METAFONT *book* are the same as corrections to Volume C. Some of the corrections below have already been made in reprintings of the books. Changes to Volume B refer to the fourth printing (1991), which differs markedly from earlier printings because it includes all the revisions for TEX3.0. Changes to Volume D refer to the third printing (1991), which differs markedly from earlier printings because it includes all the revisions for METAFONT 2.0. Changes to the mini-indexes and master indexes of Volumes B and D are not shown here unless they are not obviously derivable from what has been shown.

`errorlog.tex`

Appendix to the Errors of TEX paper (updated) Section numbers now adjusted to TEX 3.0 equivalents

`logmac.tex`

Macros for the appendix to "Errors of TEX" paper

`mf84.bug`

This file has been updated periodically ever since METAFONT84 was born. What you are about to read is "authentic source material" from the early days before the program converged. Module numbers on the first entries may bear little relation to those in Volume D.

`cm85.bug`

This file is a log of changes made to the "new generation" of Computer Modern fonts, after the first output proofs were ready. I had a great deal of advice from Neenie Billawala, Matthew Carter, and Richard Southall while I was making these revisions.

`tex82.bug`

This file has been updated periodically ever since TEX82 was born; it has been summarized in "The errors of TEX," Software Practice & Experience, July 1989. Entries are in chronological order; thus the most recent news (including all bugfixes made since that article was published) appears at the bottom of the file.

Add 519 to these entry numbers to get the corresponding number in the published article. The article also translates all module numbers to their final form; what you are about to read is "authentic source material" from the early days before TEX converged.]

First updates to the TEX82 listing published in September, 1982. (These changes were included in the original Version 0 of TEX, but they were discovered after the listing went to press.)

`cm85.bug`

This file is a log of changes made to the "new generation" of Computer Modern fonts, after the first output proofs were ready.

LaTeX3; Call for Volunteers

Michael Downes*

September 1992

This is a call for volunteers to help in the development of LaTeX3. There are many tasks needing to be done in support of the LaTeX3 project which can be worked on concurrently with the development of the LaTeX3 kernel. Furthermore, some tasks require special expertise not found among the core programming team. Initial research, analysis, and work on these tasks by volunteers can greatly speed up the process of integrating a number of desirable features into LaTeX3. Many of these features can be extensively developed and tested under LaTeX 2.09 even before the LaTeX3 kernel is available.

Therefore we are publishing a list of tasks to the LaTeX user community through various channels and we ask readers to consider contributing some time and effort (particularly, but not exclusively, readers with expertise in the various areas touched on). The task list is distributed in the form of a LaTeX article; it is fairly readable in electronic form, and it can be printed on paper if desired.

Getting and reading the LaTeX3 volunteer task list

- Anonymous FTP from the following sites.
  ```
  niord.shsu.edu
      directory [fileserv.vol-task]
  ftp.uni-stuttgart.de
      directory soft/tex/vol-task
  ```
- Mail server from
  ```
  fileserv@shsu.bitnet
  ```
  or
  ```
  mail-server@rus.uni-stuttgart.de
  ```
  Send mail to `fileserv@shsu.bitnet`. No subject line is necessary. In the body of the mail, write one line:
  ```
  sendme vol-task
  ```
  Send mail to mail-server@rus.uni-stuttgart.de. No subject line is necessary. In the body of the mail, write one line:
  ```
  send soft/tex/vol-task/vol-task.tex
  ```
- Request a printed copy from the TeX Users Group:
  TeX Users Group
  `tug@math.ams.org`
  PO Box 9506
  Providence, RI 02940
  USA
  401-751-7760
- Read it in your copy of TUGboat (or your friend's copy, or your library's copy ...) when it is published there [soon].

It will also be mailed initially to some mail lists and newsgroups that are likely sources of volunteers. Suggestions for additional publication destinations are welcome.

If after reading the descriptions you are interested in working on a particular task, the first step is to contact the coordinator for that task (if one is listed), or else contact the volunteer list manager:

George D. Greenwade
Department of Economics and Business Analysis
College of Business Administration
P.O. Box 2118
Sam Houston State University
Huntsville, Texas, USA 77341-2118
`bed_gdg@SHSU.edu (Internet)`
`BED_GDG@SHSU (BITNET)`
`SHSU::BED_GDG (THENET)`
telephone (409) 294-1266
FAX (409) 294-3712

If someone else is already serving as the task coordinator for that task, that person will discuss with you the current status of the work and ways in which you might contribute. Otherwise, the list manager will designate you as the 'task coordinator' for that task, and assist you in getting answers to any initial questions you may have. Further details are found in Appendix A of the task list.

*Extracted from LaTeX-X Mailing list:`LATEX-X@DHDURZ1.BITNET`

EuroTEX '92 proceedings

Jiri Zlatuska

October 1992

EuroTEX '92 proceedings volume contains 330 pages of papers presented at the last European TEX Conference held in Prague, Czechoslovakia.

There are full texts of five invited talks included, presenting topics ranging from the future of TEX to combinations involving METAFONT and PostScript, and also user support.
There are 20 high-quality papers by the authors of the contributed talks, presenting wide range of topics, TEX programming techniques, the use of TEX for languages with non-European alphabets, developments in the use of TEX in Eastern Europe, discussions new user environments, incorporating PostScript fonts, and issues concerning professional typesetting in national alphabets.

1 How to order

The volume is available from the Czechoslovak TEX user's group for only DM 30.– (or other local currency equivalent) payed by money transfer to the account:

```
34735-021/0100 at KOMERCNI BANKA,
                  PRAHA,
                  CZECHOSLOVAKIA
```

The address of the bank is:

KOMERCNI BANKA,
pob. Praha – MESTO,
Vaclavske nam. 42
110 00 PRAHA 1
Czechoslovakia

and the name of the account is:

```
Ceskoslovenske sdruzeni uzivatelu TEXu
```

Send the copy of your payment receipt with your order to the following address:

Ceskoslovenske sdruzeni uzivatelu TeXu (CSTUG)
Sokolovska 83
186 00 Praha 8
Czechoslovakia

Money transfer is *strongly* preferred, esp. via SWIFT. In case of problems, a personal check can be sent to the above address, but really in exceptional cases. In such a case, write the check on:

```
"Ceskoslovenske sdruzeni uzivatelu TeXu"
```

No credit card payments, though.

2 Bulk Orders

For bulk orders, from, e.g., national TeX users groups, we offer the price DM 25.– per issue. Please contact Jiri Vesely `<jvesely@cspguk11.bitnet>` or Karel Horak `<horakk@csearn.bitnet>` for arrangement. Generally, we strongly prefer this form. We hope this may provide for even better price for anybody interested, and to ensure spreding this volume containing *really* interesting material.

3 Finally

Don't miss the opportunity to order, and to receive the up-to-date volume on recent development related to TEX both in Europe and in the U.S., including exciting material on possible future development of computer typesetting in the spirit of software developed by Don Knuth.

TUG '93; Call for papers

World Wide Window on TEX

14th Annual TEX Users Group Meeting July 26th – 29th, 1993

Aston University in Birmingham, UK, will be the venue for the 1993 TUG conference. Aston is the home of the 'Aston Archive', one of the largest collections of electronic TEX paraphernalia. This is the first time the annual meeting will have been held outside North America.

The location of the conference at one centre of the electronic web and its movement from North America encourages particular focus on the 'world-wide' aspects of TEX (LATEX, METAFONT...). The marked rise in maturity of windowing systems (Macintosh, Atari, Amiga, Windows3, X windows) also allows us to exploit more straightforward and direct ways of employing the TEX tools. It is hoped that there will be a contribution to the conference from the Didot project, further extending the range of topics to include digital typography and font creation.

The conference will feature the regular paper presentations, but workshops, poster displays, courses, panels and 'birds of a feather' sessions will be integral components.

Contributions are being actively sought in the following subject areas: ◇ archives ◇ electronic networks ◇ formatting structured documents ◇ LATEX3 ◇ graphical user interfaces to TEXware ◇ non-english issues ◇ non-Latin scripts ◇ digital typography ◇ editing structured documents ◇ styles ◇ other typesetting systems ◇ document views ◇

Program coordinators

	Chris Rowley	Malcolm Clark
	Parsifal College	IRS
	Open University	University of Westminster
	Finchley Road	115 New Cavendish Street
	London NW3 7BG	London W1M 8JS
phone:	071 794 0575	071 911 5000 ex 3622
email:	ca_rowley@uk.ac.open.acs.vax	malcolmc@uk.ac.wmin
fax:	071 433 6196	071 911 5093

Conference committee

Peter Abbott, Chris Rowley, Philip Taylor, Carol Hewlett, Sebastian Rahtz, David Osborne, Malcolm Clark

Table of Contents TUGboat

Volume 13.2 and 13.3

July 1992 / October 1992

TUGboat tables of contents files are on `math.utah.edu` in `pub/tex/pub/tugboat`, also accessible via `tuglib@math.utah.edu` server by 'send index from tex/pub/tugboat'.

1 TUGboat 13.2 (July 1992)

2 TUGboat 13.3 (October 1992)[1]

[1] 1992 Annual Meeting Proceedings