Nederlandstalige TEX Gebruikersgroep

Aan : Leden Nederlandstalige TEX Gebruikersgroep,
: Secretariaten Internationale TEX Gebruikersgroepen.

Betreft : Verslag 8^e NTG bijeenkomst

Locatie : Technische Universiteit Eindhoven
Datum : 21 november 1991
Behandeling : bij de volgende bijeenkomst (4 juni 1992 te Amsterdam)

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:
H.P.A. Mulders,
KUB, Tilburg.
Internet: s172hmul@htikub5

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 : E. Algera (EGD); P. Bloemen (TUE); J. Braams (PTT Neher Laboratorium); L. Combee (TUD); L. de Coninck (de Kraal); P. Derks (Howden Food); M. Dings (AKZO); R. Doornebal (Wolters Kluwer); S. Eggermont (TUE); E.J. Evers (RUU); E. Frambach (RUG); J.B.W. Hoebeek (TUD); C.F.A. Janssen; T.A. Jurriens (RUG); R. Kappert (KUN); H. Kołodziejska (MacroSoft Ltd, Polen); C.G. van der Laan; A. de Leeuw van Weenen (RUL); D.A. van Leeuwen (RUL); H. van der Meer (UvA); A.J. de Meijer (RUU); H.P.A. Mulders (KUB); G.J.H. van Nes (ECN); G. Oomen (Wolters Kluwer); D. Taupin (Univ. Orsay, Frankrijk); P. Tutelaers (TUE); E. Ulijn (TUD); P. Vanoverbeke; E.J. Vens (RUG/ICCE); J. van Weerden (RUU); F. van de Wiel (CWI); J.J. Winnink;

Notulisten : G.J.H. van Nes & J.J. Winnink

1 Opening

Om ongeveer 10:15 uur opent voorzitter van der Laan de bijeenkomst en heet een ieder van harte welkom. In het bijzonder heet hij de twee buitenlandse gastsprekers Hanna Kołodziejska en Daniel Taupin welkom. Deze bijeenkomst heeft *'Fun with TEX'* als thema en dat zal 's middags tijdens een aantal lezingen uit de doeken worden gedaan.

Het ledental van de NTG bedraagt inmiddels 150 waaronder 30 instituutsleden. Dit is ongeveer twee keer zoveel als twee jaar geleden werd verwacht.

Voor het voorjaar van 1992 wordt een bijeenkomst voorbereid met als titel *'TEX and Scientific Publishing'*. Voor de najaarsbijeenkomst zal het onderwerp zijn *'Typografie'*[1].

Voor de komende MAPS zijn inmiddels enkele artikelen binnen. De redactie bestaat nu uit Gerard van Nes, Erik-Jan Vens en Jos Winnink. Victor Eijkhout en Nico Poppelier hebben in het verleden hun medewerking in de een of andere vorm ook toegezegd.

Het wederzijds lidmaatschap tussen NTG en TUG is rondgekomen. Nader bericht erover zal nog worden rondgestuurd. Er geldt een korting van ongeveer 10%. Daarnaast hoeft een gemeenschappelijk lid nu slechts één rekening te voldoen.

Naast dit goede nieuws is er ook slecht nieuws en wel met betrekking tot TUG zelf. De financiële situatie van TUG is nog steeds niet rooskleurig en voor 1991 bedraagt het tekort $ 1500. Het gevolg hiervan is dat de nummers 3 en 4 van *TUGboat* van dit jaar zijn vertraagd en in hun geheel gewijd zullen zijn aan de proceedings van Dedham.

Buiten de Verenigde Staten is de situatie van de TEX-gebruikersgroepen goed. Zo bestaan er inmiddels zo'n tien groepen waaronder in Polen (gestart door Hanna Kołodziejska), Tjechoslowakije, Rusland en Japan.

Gewezen wordt op de nu aanwezige tafel met te koop zijnde TEX-achtige boeken. Een korting van 10% voor NTG leden geldt. Ondanks de enigszins beperkte keuze blijkt deze boekenverkoop een duidelijk succes te zijn; zeker voor herhaling vatbaar.

2 Verslag bijeenkomst 2 mei 1991

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

3 Ingekomen stukken en Mededelingen

Enkele tijdschriften van zusterverenigingen zijn wederom ontvangen. Deze liggen op de leestafel ter inzage.

De volgende mededelingen worden gedaan:

- **Contact met wiskundigen**
  Er is een bijeenkomst geweest met CWI, NTG en het Wiskundig Genootschap. Tijdens deze bijeenkomst zijn enkele resultaten geboekt. NTG heeft 1 pagina ruimte gekregen in *'Mededelingen van het Wiskundig Genootschap'*, voor het geven van voorbeelden van TEX gebruik en het promoten van de NTG e.d. Gesproken is verder over de TEX cursus in juni 1992. Daarnaast subsidieert het Wiskundig Genootschap de komst van Ralp Youngen i.v.m. de komende NTG voorjaarsbijeenkomst.
- **Atari gebruikersgroep**
  De Atari groep vraagt medewerking bij een publikatie over TEX in een tijdschrift (ST tijdschrift). Erik-Jan Vens heeft met deze groep contacten. Ook

Editors van deze MAPS zijn: G.J.H. van Nes, E.J. Vens & J.J. Winnink.

Het verslag van de NTG bijeenkomst op 21 november 1991 is (in concept) eind maart 1992 via e-mail dan wel via de post reeds toegestuurd naar alle NTG leden.

[1] Onlangs is besloten dit onderwerp naar de NTG lustrum bijeenkomst in het voorjaar van 1993 te verschuiven. Het thema van de najaarsbijeenkomst zal zijn: 'The future of TEX & LATEX'

heeft deze groep een PD-versie van TEX in de aanbieding. Wat hiervan de specificaties zijn is nog onbekend.

- **Welkomst pakket/folder**
  Huub Mulders wacht nog steeds op enkele bijdragen. Omdat de situatie toch wel dringend wordt en we niet nog al te lang kunnen blijven wachten, wordt besloten om het huidige materiaal reeds tot een pakket te verwerken. Onderwerpen die ondermeer opgenomen zullen worden zijn: TEX en wiskunde, schaken, muziek, bridge, en later mogelijk tabellen en astronomie. Ook het artikel van Hoenig is interessant. Verwachting is dat het pakket in de loop van de tijd zal groeien. Een aantal gebieden zoals Chemie, Fysica en Letteren moeten echter niet vergeten worden.
- **Begroting van 1992.**
  ↪ *Voorgesteld en vervolgens aangenomen wordt het voorstel om de begroting in de najaarsbijeenkomst te behandelen, daar dit beter uitkomt dan op de voorjaarsbijeenkomst in mei/juni* .
  Bij de begroting van 1992 (zie MAPS 91.2; bijlage C) zijn geen opmerkingen. Over de financiële situatie van 1991 valt op te merken dat de verwachte intering op het eigen kapitaal niet heeft plaatsgevonden daar door de onverwacht snelle groei van het aantal leden de inkomsten zo snel stijgen dat de NTG quite speelt.
  Op de vraag van Jurriens of er geen post op de begroting is opgenomen ten behoeve van het verspreiden van PD TEX implementaties (kosten *f* 1,- tot *f* 1,50 per diskette?), wordt geantwoord dat deze suggestie van 'professionele kopieerservice' in gedachten zal worden genomen.

4 Verslag/discussie werkgroepen

4.1 Werkgroep 1: Educatie

De bijdragen in de MAPS (91.2) geven geen directe aanleiding tot discussie.

Van der Laan deelt mede dat er op dit moment gewerkt wordt aan een vijf dagen durende TEX cursus voor een ledenprijs (= bodemprijs) van *f* 100. Docent is David Salomon. Voor deze cursus zijn er op dit moment reeds zo'n 15 à 20 gegadigden. Gestreefd wordt naar ongeveer 50 cursisten. Cursusmateriaal is bij van der Laan reeds aanwezig. Betreffende de cursusruimte wordt gehoopt op een donatie van het CWI. Indien de cursus doorgaat zal dit een duidelijk succes zijn binnen de TEX wereld. Vooral daar de laatste TEX cursus in Dedham slechts 5 cursisten trok.

Er wordt opgemerkt dat vijf dagen wel erg lang is, hetgeen potentiële deelnemers zou kunnen afschrikken. Daarom zal in de komende aankondigingen duidelijk worden gemaakt dat de cursus bestaat uit vijf min of meer disjuncte aandachtsgebieden en dat het derhalve ook mogelijk is om slechts aan een gedeelte van de cursus deel te nemen.

4.2 Werkgroep 7: PC-zaken

Winnink deelt mede dat de situatie nu zodanig is dat op basis van emTEX een NTG-distributie versie kan worden gemaakt voor MS-DOS systemen, waarbij op dit moment alleen nog wordt gezocht naar een geschikt spellingscontrole programma. ISPELL werkt, doch in sommige situatie treedt er een crash van het systeem op. De distributieversie komt beschikbaar inclusief installatieprocedure. Van emTEX is een β-release van 3.14 aanwezig. Een spellingchecker die onlangs door Jan van der Steen is gemaakt schijnt beter te zijn.

Hoe gemakkelijk de gewenste combinatie ook voor andere platforms te maken is, is gezien de magere reacties (lees: belangstelling) ervoor tot nu toe, niet vast te stellen. Op dit moment krijgt de MS-DOS versie voorrang zonder de weg naar andere platforms af te sluiten. Er is wel een min of meer komplete PD Atari versie beschikbaar Voor de Macintosh schijnt er ook een volledige versie te zijn. Vens heeft voor belangstellenden een Atari en een Macintosh (OzTEX) implementatie bij zich.

Vens deelt mede dat SbTEX zeer goed bij de Fransen aanslaat. Taupin geeft dit toe, doch hij prefereert echter emTEX vanwege bepaalde voordelen met deze implementatie. SbTEX is op zich wel goed (sluit ondermeer aan bij de Franstalige typesetting), doch kan na enige tijd (uren) werken wel eens crashen. De emTEX implementatie werkt zelfs bij 390kb vrij RAM geheugen.

Tutelaers deelt mede dat de TUE sinds kort over is gegaan van PCTEX naar emTEX. Gedistribueerd wordt een originele set plus locale toevoegingen plus installatie script. Daarnaast hebben zij een procedure voor een minimale TEX installatie ('verbruikt' minder dan 3.5 Mb). Winnink zegt toe dat voor de officiële NTG-emTEX distributie van deze kennis gebruik gemaakt zal worden.

Genoemd wordt een specifieke userinterface voor emTEX die door Eberhard Mattes locaal wordt gebruikt. Informatie wordt uitgewisseld hoe de emTEX implementatie te verkrijgen is.

4.3 Werkgroep 8: Nederlandstalige TEX gebruikersdag

Er zweven nog een aantal zaken:

1. De decharge van de commissie die de SGML/TEX dag van 1990 heeft georganiseerd.
   Er kunnen geen financiële gevolgen meer worden verwacht met betrekking tot de NTG-dag van 1990. De commissie wordt vervolgens door de vergadering gedechargeerd.
2. Wat moeten we met de NTG-dag?
   Wat is de plaats van de NTG-dag in het zicht van de Europese bijeenkomsten? Het voorstel om de NTG-bijeenkomsten wat uit te bouwen (zoals de bijeenkomst van vandaag bijvoorbeeld) vindt ondersteuning. Dit zal er toe leiden dat er voorhands geen NTG-dagen meer worden georganiseerd. Ook wordt voorgesteld om te proberen de NTG eens de Europese TEX-bijeenkomst te laten organiseren.

Dit brengt ons bij het volgende punt.

3. Zijn er mensen om activiteiten te organiseren?
   Wie organiseert het? Organisatie van evenementen kost veel tijd en wie o wie kan deze tijd er in steken? Wat is er geworden van de geplande bijeenkomst in Tilburg? Deze bijeenkomst werd in eerste instantie georganiseerd door een vakgroep van de KUB en werd daarmee veel breder van opzet dan de NTG voor ogen stond. Ook was het geen *echte* NTG-dag geworden op deze manier. Conclusie: in 1992 geen NTG-dag.
   Wel zal de NTG aanwezig zijn op het mathematisch congres en op het 'Dag van het Document'. Aan beide bijeenkomsten welke begin 1992 plaatsvinden, zal een bijdrage worden geleverd. Ook is de NTG aanwezig geweest op het KNCV symposium 'MOLECUUL MUIS MANUSCRIPT' in oktober.

4.4 Werkgroep 13: Nederlandstalige TEX

Eijkhout is nog steeds coördinator. Het verhaal van David van Leeuwen over de IJ-ligatuur wordt mondiaal gedropt. Deze bijdrage is zijn persoonlijke visie. WG 13 komt met een voorstel.

4.5 Werkgroep 15: TEX 3.0

Er is geen nieuws te melden door deze werkgroep. De stand van zaken met betrekking tot LATEX 3.0 is op dit moment niet duidelijk. Er komt in november/december een nieuwe versie van LATEX 2.09. Deze versie heeft als eigenschappen:

- dat zij beter met het nieuwe fontselectie schema kan omgaan,
- de vastgecodeerde namen in style files verwijderd zijn zodat alles veel flexibeler is geworden à la BABEL,
- beter overweg kan met faciliteiten van TEX 3.

LATEX is weinig flexibel. Hans van der Meer heeft de PICTURE ENVIRONMENT veranderd op basis van EPIC. Hij zegt toe dit aan Johannes Braams op te sturen ter beoordeling.

De opmerking wordt gemaakt dat LATEX 3 modulair moet worden.

Ook in de Verenigde Staten is er inmiddels een discussie aan de gang met als richtlijn om te onderzoeken wat er is blijven liggen in TEX en niet alleen te kijken naar wat er nog allemaal mogelijk zou zijn.

5 Rondvraag

- **Van Leeuwen** noemt het bestaan van een macropakket voor TEX met de naam PhysTEX, bestemd voor gebruik door natuurkundigen. Opgemerkt wordt dat Erik van Herwijnen (CERN) veel aan TEX doet, maar dat zijn afdeling inmiddels is ingekrompen tot één persoon, zijnde hemzelf. Vermeld wordt tevens dat de *European Physical Society* bezig is met het maken van een gestandaardiseerde style file voor natuurkundigen. Ook de Europese uitgevers van tijdschriften voor astronomen proberen tot een standaard te komen. Vermoedelijk zal dit bij hun LATEX worden.
  Van Leeuwen merkt op dat de kennis van TEX en LATEX zijns inziens te beperkt is. Er moet iets komen zoals *'Math into Blues'*. Templates zouden aanwezig moeten zijn. Het probleem ligt in de coördinatie van de informatie en de uiteindelijke uitvoering. Ook zou eens contact met de Nederlandse Natuurkundigen (met hun uitgave: 'Nederlands Tijdschrift voor Natuurkunde') opgenomen moeten worden. Oomen (Wolters Kluwer) merkt op dat het in de praktijk volgens hem allemaal wel mee valt, althans, dat is zijn ervaring.
  Geopperd wordt om te proberen een bijdrage m.b.t. de TEX zaken te plaatsen in de bijlage 'Onderwijs & Wetenschap' van de NRC. De mening is verdeelt of zo'n artikel wel geaccepteerd wordt. I.h.a. blijft het moeilijk om mensen te interesseren om iets over TEX op te nemen in hun tijdschriften/kranten. Zo zijn bijvoorbeeld de contacten met SURF op niets uitgelopen.
  Theo Jurriens merkt op dat hij gevraagd is om op zeer korte termijn een overzichtsartikel te maken voor een astronomisch tijdschrift. Mogelijk heeft de NTG daar wat aan.
- **Evers** meldt dat op de leestafel een ruw overzicht (met datum, afzender, onderwerp) ligt van de electronic mails die gedurende afgelopen jaren op de discussielijst TEX-NL zijn geplaatst.
- **Van der Laan** deelt mee dat er op dit moment gediscussieerd wordt over de vraag of TEXHAX moet blijven voortbestaan. Hij vraagt om ideeën zodat deze via hem naar de betreffende werkgroep doorgestuurd kunnen worden.
- **Janssen** vraagt naar de beschikbaarheid van de Nassi-Schneidermann en Flow-Chart stylen.
- **Eggermont** merkt op dat WEAVE en TANGLE in de PC distributie niet goed samenwerken met Turbo Pascal. Geopperd wordt om de source ergens anders vandaan te halen dan wel om contact met de auteurs op te nemen.
- **Dings** vraagt of een WEB naar MODULA2 converter bestaat. Geantwoord wordt dat deze in OZTEX zou moeten zitten. Ook mogelijk bij de oude UNIX distributie. Vens zou proberen de software vanuit een fileserver op te halen.
- **Jurriens** suggereert om ook de beroepsverenigingen (wiskundigen, fysici, chemici) te benaderen voor de voorjaarsbijeenkomst van 1992 door ze bijvoorbeeld een aankondiging te sturen.
  Geantwoord wordt dat eerst de resultaten van de benadering van de mathematici moeten worden afgewacht. Het is echter nooit weg om de andere verenigingen eventueel te informeren.

6 NTG presentaties: 'Fun with TEX'

Zes presentaties worden gedurende de middag gegeven, t.w.:

- *'Tower of Hanoi'*, door Kees van der Laan;
- *'Dating with TEX'*, door Theo Jurriens (Rijksuniversiteit Groningen);
- *'Chess with TEX and LATEX'*, door Piet Tutelaers (Technische Universiteit Eindhoven);
- *'Bridge with TEX and LATEX'*, door Kees van der Laan;
- *'Go with TEX'*, door Hanna Kołodziejska (MacroSoft Ltd, Polen);
- *'Music with TEX'*, door Daniel Taupin (Universiteit Orsay, Frankrijk).

De inhoud van genoemde lezingen zijn opgenomen in MAPS 91.2 en MAPS 92.1. Van de lezing 'Music with TEX' worden overdrukken op de vergadering uitgedeeld. De macro's behorende bij bovengenoemde onderwerpen zijn ondermeer te verkrijgen via de TEX-NL fileserver in Nijmegen dan wel via de fileserver van de Rijksuniversiteit Utrecht (zie MAPS 90.2 bijlage G). De nu reeds zeer goed bruikbare schaak en muziek macro's blijken nog steeds verder ontwikkeld te worden.

7 Sluiting

De gastheer Piet Tutelaers en het Rekencentrum van de Technische Universiteit Eindhoven worden bedankt voor hun gastvrijheid. De aanwezigen en in het bijzonder de buitenlandse gasten worden bedankt voor hun bijdragen.

De volgende vergadering is op:

donderdag 4 juni 1992

bij het Centrum voor Wiskunde en Informatica (CWI) te Amsterdam; gastheer Gerrit Stemerdink. Het thema is 'TEX and Scientific Publishing'.

De vergadering wordt rond 17:00 uur gesloten.

Getekend:
Voorzitter: Secretaris:

TEX kalender 1992

4	jun	NTG (9^e)	CWI, Amsterdam
15–19	jun	**NTG-TEX course** [1]	RUG, Groningen
16–18	jun	GUTenberg ’92	Parijs
27–30	jul	TUG’92	Portland, Oregon, USA
14–18	sep	EuroTEX’92	Praag, Tsjecho Slowakije
19	nov	NTG (10^e) [2]	Meppel

Glossary

AMS	:	American Mathematical Society
BoD	:	Board of Directors
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)
ITALIC	:	LUG Ierland
Nordic	:	LUG Scandinavië, Denemarken, en IJsland
NTG	:	LUG Nederland en Belgie
UKTUG	:	LUG Engeland
YUNUS	:	LUG Turkije (feitelijk alleen nog een discussielijst)
GUST	:	LUG Polen
TTN	:	TEX and TUG News
TUGboat	:	Magazine TUG
MAPS	:	Minutes and APpendiceS
SGML	:	Standard Generalized Markup Language
lxiii	:	LATEX 3.0

Aanschaf TEX boeken door NTG leden

Net voor het drukken van deze MAPS werd met de uitgeverij Addison-Wesley overeengekomen dat NTG leden een korting ontvangen van ongeveer 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 een extra korting betekend voor de NTG leden.
De aanbieding geldt echter tot eind juni 1992.

De volgende voorwaarden gelden:

1. Bestelling direct bij Addison-Wesley t.a.v. Rita Snaddon persoonlijk.
2. Betaling per credit card (andere mogelijkheden zijn er niet).

Behalve de TEX boeken, geldt deze regeling ook voor een aantal boeken over PostScript die door de uitgever worden uitgegeven.

Meer informatie is te vinden in de folder die bij deze MAPS is bijgesloten.

[1] Voor meer informatie zie de betreffende bijlage elders in deze MAPS.
[2] Onderwerp: ‘The future of TEX & LATEX’; sprekers ondermeer Frank Mittelbach en Yannis Haralambous.

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.)
   E.J. Vens (RUG) *
   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)
   G.J. Braas (EGD; Archimedes)
   H. Brouwer (EGD)
   P. Tutelaers (TUE)
   E.J. Vens (RUG; DOS)
   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) *
    E.J. Vens (RUG)

* coördinator

Van de Voorzitter

Maart 1992

1 NTG

De contacten met de wiskundigen komen van de grond: stukjes in de 'mededelingen,' subsidie reis Ralph Youngen, bijdrage op wiskunde congres over TEXing Math. Het CWI is instituutslid. Who next? De astronomen, de chemici, de fysici, ...? Of alle drie tegelijk?

PR materiaal is beschikbaar (info-pakket, ontworpen op de groei). PD distributie van o.a. emTEX moet nog wat professioneler.

Educatie is en blijft een belangrijk aandachtspunt. Dit jaar de unieke 'Insights and Hindsights' cursus[1] Bovendien wordt David's cursusmateriaal gepubliceerd.

Het lxiii (LATEX-3.0) project wordt door de NTG gesubsidieerd!

De komende bijeenkomst heeft als thema 'TEX and Scientific Publishing,' met mogelijk een workshop de volgende ochtend: 'Mathematicians needs in publishing.' Uit Amerika nemen deel Ralph Youngen van AMS en David Salomon.

Het najaar staat in het teken van LATEX: de ontwikkelingen, fontselectie, en een beetje los daarvan de virtuele fonts. Frank Mittelbach, Yannis Haralambous, en Jiri Zlatuska hebben hun medewerking toegezegd.

Voorjaar '93 vieren wij ons lustrum onder het thema 'Typografie'.

2 LUGs

DANTE maakt zich sterk in de discussie over de Toekomst van TEX.
GUST (de Poolse TEX gebruikersgroep) organiseert zich (Bravo Hanna!).
Verschillende GOS staten slaan de handen ineen, met als paraplu CyrTUG. Watch out, meer, veel meer, staat er aan te komen!
Ondanks een halve wereldbol aan verschil hoort Japan er helemaal bij.

3 TUG

TUG is organisatorisch topzwaar. De office is geslonken, dat wel. Desalniettemin zijn de financiën nog niet rooskleurig. TTN floreert, en TUGboat gaat maar door. Prima! De TEXniques serie zou wel een facelift en in het algemeen een enthousiaste injectie—lees redactie—kunnen gebruiken. Er is $18.000 uitgetrokken voor 6 nummers!

De BoD is verkozen, met Nico Poppelier erin. Het dagelijks bestuur voor 1992 bestaat uit Malcolm Clark, voorzitter, Kenneth Dreyhaupt, vice-voorzitter, Peter Flynn, secretaris, en Bill Woolf, penningmeester.

4 Voorjaarsgedachten

Energiestromen. Hoe zit het daarmee bij de TEXies?
Er zijn en aantal belangrijke activiteiten: lxiii, LAMS-TEX, eplain, ..., naast Hoenig's METAfont-TEX interactie, de pennevruchten van auteurs van boeken, courseware en artikelen, en het achter de schermen opereren van hen die het leven in de brouwerij houden. Is er verspilling? Spelen wij wel goed op elkaar in? Organisatorisch zijn er wat dwarsverbanden. TEXnisch gesproken gaat er kennelijk nogal wat bij elkaar langs.

Ik vind het bijvoorbeeld vreemd dat er gefocusseerd wordt op lxiii, terwijl o.a. Spivak, Hendrickson, en Berry al met projecten bezig waren in de sfeer van procedurele markup, ter verbetering van LATEX, of algemener ter verbetering van (La)TEX gebruik. Let wel dat was al gaande ruim voor 1989, het geboortejaar van lxiii.

Competitie en samenwerking, het blijft een merkwaardig duo. Ik hoop echter dat lxiii wel kennis neemt van de goodies van o.a. LAMS-TEX.
Ondanks de normale intermenselijke spanningsvelden gaan de groeperingen redelijk goed met elkaar om: nieuws wordt gedeeld, activiteiten wederzijds gestimuleerd, en wij komen mondjesmaat bij elkaar over de vloer.

En hoe zit het met de buitenwereld? Met al die activiteiten zoals aangestipt op de 'Dag van het Document'? Wat gebeurt er bij Adobe, bij Rank Xerox? In hoeverre heeft de DTP wereld de markt veroverd? Waar staan de Wordperfecters? Hoe is het met SGML? Heeft die stroming zijn belofte waar kunnen maken? Vragen, en nog eens vragen.

Laten wij gewoon maar eerlijk en bescheiden doorgaan, en goed voor de geest houden waar het allemaal om draait, waar het spul goed in is, en waarin niet.
Het voorjaar blijft gewoon zijn werk doen, alles schiet de grond uit.

[1] Gesubsidiëerd door NTG, geen financiële drempel dus!

Jaarverslag NTG

jan–dec 1991

Gerard van Nes

1 Algemeen

In 1991 ging de NTG zijn 4^e jaar van bestaan in. Wederom verschenen er een tweetal MAPS (Minutes & APpendiceS), twee NTG bijeenkomsten vonden plaats met een groeiend aantal lezingen, bestuursverkiezingen werden voor het eerst gehouden, een gecombineerd NTG/TUG lidmaatschap werd mogelijk, de samenwerking met Wiskundig Nederland kreeg gestalte, en er werd verdere aandacht besteed aan de continuering en acceptatie van de NTG.

2 Het NTG bestuur

Het NTG bestuur bestond uit de volgende personen:

- C.G. van der Laan, voorzitter
- G.J.H. van Nes, secretaris
- J.L. Braams, penningmeester
- H.P.A. Mulders, bestuurslid
- T. de Klerk, bestuurslid (tot mei 1991)
- J.J. Winnink, bestuurslid (vanaf mei 1991)

Op de bijeenkomst in mei vonden voor het eerst bestuursverkiezingen plaats. T. de Klerk en C.G. van der Laan traden reglementair af, waarbij laatstgenoemde zich herkiesbaar stelde. De door het bestuur voorgestelde twee kandidaten, C.G. van der Laan en J.J. Winnink, werden bij acclamatie gekozen.

3 Het NTG ledenbestand

Het ledenaantal steeg in 1991 langzaam doch duidelijk. Eind 1991 telde de NTG 159 leden waarvan 28 instituutsleden. T.o.v. eind 1990 betekent dit een stijging met ruim 40 leden (waaronder 7 instituutsleden). De contributie bleef in 1991 ongewijzigd.
De volledige NTG ledenlijst met aanvullende hardware en software informatie werd wederom gepubliseerd in de MAPS (# 91.1 en # 91.2).

4 De NTG bestuursvergaderingen

Op 25 januari en 1 oktober vonden telefonische NTG bestuursvergaderingen plaats waarin naast de algemene gang van zaken binnen de NTG, de organisatie NTG dagen, NTG/TUG lidmaatschap, en organisatie NTG bijeenkomsten ter sprake kwam. Daarnaast werd aandacht besteed aan ondermeer bestuursverkiezingen en de NTG toekomststrategie.

5 De NTG bijeenkomsten

Er werden in 1991 twee NTG bijeenkomsten georganiseerd welke gekenmerkt waren door een levendige discussie en een hoge mate van informatieuitwisseling:

1. Op 2 mei 1991 bij Elsevier Science Publishers te Amsterdam. Aanwezig waren 40 leden.
2. Op 21 november 1991 bij de Technische Universiteit Eindhoven. Aanwezig waren 32 leden.

Op deze bijeenkomsten vonden de volgende lezingen plaats:

- ‘Gebruik van TEX binnen het EGD’ door J.A. Jager en P. Sader,
- ‘Gebruik van TEX en LATEX op het CAWCS’, door het echtpaar van Geest,
- ‘Tower of Hanoi’, door C.G. van der Laan,
- ‘Dating with TEX’, door T.A. Jurriens,
- ‘Chess with TEX and LATEX’, door P. Tutelaers,
- ‘Bridge with TEX and LATEX’, door C.G. van der Laan,
- ‘Go with TEX’, door H. Kołodziejska,
- ‘Music with TEX’, door D. Taupin.

Van bovengenoemde twee vergaderingen verschenen uitgebreide verslagen met bijlagen (MAPS # 91.1 en 91.2).

Het plaats laten vinden van een bijeenkomst met een vooraf bepaald thema (in 1991: ‘TEX/LATEX gebruik bij een bedrijf’ en ‘Fun with TEX’), bleek duidelijk een succes te zijn. Een tweetal lezingen werden gegeven door buitenlandse sprekers.
De leestafels met TEX-achtige boeken, tijdschriften, brochures en andere aan TEX verwante documenten, trok wederom de nodige belangstelling. Bij de laatst genoemde bijeenkomst was het tevens mogelijk om TEX/LATEX boeken met korting aan te schaffen.

6 De NTG MAPS

Ook in 1991 verschenen er wederom twee MAPS met in totaal ruim 270 goed gevulde bladzijden waarin bijdragen van diverse werkgroepen, verslagen van nationale en internationale bijeenkomsten, book reviews, software reviews, artikelen van NTG lezingen (TEX en schaak, bridge, Go), server informatie, algemene TEX

en LaTeX software overzichten, artikelen over locaal TeX en LaTeX gebruik, educatiemateriaal, technische bijdragen etcetc. Het uiterlijk van de MAPS kreeg steeds vastere vormen aan, terwijl de redactie eind 1991 werd uitgebreid van één naar drie personen.
Zowel de kwaliteit als de kwantiteit van de MAPS was in een duidelijke opgaande lijn.

7 De NTG werkbezoeken

Namens de NTG werden in 1991 de volgende bijeenkomsten bezocht:

- TUG91 bijeenkomst te Dedham/USA.
  Van 15–18 juni werd door de voorzitter deelgenomen aan deze algemene TUG bijeenkomst.
- TUG BOD meeting te Dedham/USA.
  Aan deze 'Board of Directors' meeting nam namens de NTG eveneens de voorzitter deel.
- EuroTeX '91 te Parijs.
  Van 23–26 september werd ondermeer door de voorzitter en de penningmeester deelgenomen aan deze Europese TeX gebruikersbijeenkomst.
- Symposium Chemische Tekstverwerking te Amsterdam.
  Op 18 oktober werd door de secretaris deelgenomen aan het symposium 'Molecuul Muis Manuscript', georganiseerd door de KNCV, sectie Computertoepassingen.

Daarnaast bezocht de voorzitter tijdens zijn verblijf in Dedham, het AMS- en het TUG bureau.

Van de eerste drie genoemde bezoeken is door de voorzitter uitgebreid verslag gedaan in MAPS # 91.2.
Van de vierde genoemde bijeenkomst is een verslag opgenomen in MAPS # 92.1.

8 De NTG werkgroepen

Een aantal werkgroepen waren wederom in 1991 actief tot zeer actief. Dit resulteerde ondermeer in het (her)beschikbaar stellen c.q. distribueren van de laatste versie van de uitgebreide en uitstekende DOS TeX/LaTeX implementatie emTeX (nu algemeen gebruikt door de NTG leden), onderzoek en verspreiding van specifieke Nederlandstalige TeX zaken (waaronder het 'babel' systeem), en onderzoek/ondersteuning op het gebied van TeX communicatie (waaronder het TEX-NL beheer).
In de beide MAPS van 1991 verschenen diverse bijdragen van de werkgroepen.

9 De nieuwe NTG bestuursactiviteiten

- De contacten met de Nederlandse Wiskundigen zijn in 1991 verstevigd middels een bespreking op 17 oktober met het CWI en het Wiskundig Genootschap. Dit resulteerde ondermeer in de toezegging om een vaste rubriek te starten in het tijdschrift 'Mededelingen van het Wiskundig Genootschap'.
- De vele besprekingen die afgelopen jaren zijn gehouden met TUG om te komen tot een gemeenschappelijk NTG/TUG lidmaatschap kwamen eind 1991 tot een resultaat: een NTG/TUG lidmaatschap kon voor 1992 aan de NTG leden aangeboden worden met een contributiereductie van ongeveer 10% op het totale NTG/TUG lidmaatschapsbedrag.
- Een welkomspakket, bestaande uit enige informatie wat er zoal met TeX/LaTeX gedaan kon worden, kwam gereed.
- Activiteiten zijn gestart voor het organiseren van een algemene en uitgebreide TeX cursus. Deze wordt gehouden in juni 1992 te Groningen. Docent is de Amerikaan David Salomon.

10 Diversen

- De enigszins negatieve ontwikkelingen bij de TUG werden nauwlettend gevolgd. De voorzitter had zitting binnen de Boards of Directors van TUG. Functies werden door hem tevens vervuld binnen de TUG 'Long-Range Planning committee', de 'Education committee' en de 'Publications committee' (bij de laatste als voorzitter). Goede contacten zijn er tussen de NTG en de andere Europese TeX gebruikersgroepen.
- Van zowel de TEX-NL fileserver als de TEX-NL listserver werd in 1991 wederom veel gebruik gemaakt. Het aantal abonnees van de listserver steeg van 126 naar 158. Gedurende 1991 werden in totaal 846 mails verstuurd. Daarnaast werd via de NTG, diverse bestanden op de TEX-NL fileserver geplaatst.
- Vele leden maakten via e-mail dan wel via ftp dankbaar gebruik van de RUU fileserver van Piet van Oostrum voor het verkrijgen diverse soorten TeX materiaal.
- Door Theo Jurriens werd via cursussen, LaTeX in Rusland geïntroduceerd.
- Door de uitbouw van de NTG bijeenkomsten en de directe mogelijkheid om de Europese bijeenkomsten bij te wonen, werd besloten om de jaarlijkse NTG-dagen voorlopig niet meer te organiseren.
- Indirect werd medewerking verleend aan het LaTeX 3.0 project van Frank Mittelbach.
- Door NTG leden werd een belangrijk aantal artikelen gepubliceerd in TUGboat, het internationale tijdschrift van de TeX Users Group:
  - Johannes Braams, Victor Eijkhout & Nico Poppelier (1991): The Dutch national LaTeX effort, TUGboat 12#1, 21–24.
  - Victor Eijkhout (1991): The document style designer as a separate entity, TUGboat 12#1, 31–34.
  - Kees van der Laan (1991): SGML (,TeX and ...), TUGboat, 12#1, 90–104.
  - Nico Poppelier (1991): SGML and TeX in scientific publishing, TUGboat 12#1, 105–109.

- Nico Poppelier (1991): Review of 'LaTeX for engineers and scientists', TUGboat 12#2, 235–236.
- Victor Eijkhout (1991): The bag of tricks, TUGboat 12#2, 260.
- Victor Eijkhout (1991): Oral TeX, TUGboat 12#2, 272–277.
- Nico Poppelier (1991): A comment on the LaTeX column, TUGboat 12#2, 285–286.
- Johannes Braams (1991): Babel, a multilanguage style-option system for use with LaTeX standard document styles, TUGboat 12#2, 291–301.
- Victor Eijkhout (1991): Response to Paul Abrahams, TUGboat 12#2, 303.
- Nico Poppelier (1991): Two sides of the fence, TUGboat 12#3, 353–358.
- Kees van der Laan (1991): Math into BLUes, TUGboat 12#4, 485–501.

- Victor Eijkhout publiceerde het boek 'TeX by Topic' (zie bespreking elders in deze MAPS).
- Victor Eijkhout was editor van de macro column van de TUGboat.
- Nico Poppelier was lid van het 'Knuth scholarship committee'.
- Door Johannes Braams, Kees van der Laan en Theo Jurriens zijn lezingen gegeven op de EuroTeX 91 conferentie te Parijs.

  Op de TUG91 bijeenkomst in Dedham/USA trad Nico Poppelier op als keynote spreker.

Financieel verslag 1991
Nederlandstalige TEX Gebruikersgroep

Johannes Braams

Maart 1992

1 Inleiding

Voor U ligt het financieel jaarverslag van de NTG vereniging over het jaar 1991. Het jaar 1991 is het eerste jaar waarover een volledig verslag kan worden gemaakt inclusief begin- en eindbalans. De financiële ontwikkeling van de vereniging wordt aan de hand van de winst en verliesrekening en de balans besproken.

2 De winst en verliesrekening

In tabel 1 is de winst- en verliesrekening over 1991 weergegeven. De diverse posten worden in de volgende paragrafen kort toegelicht. Ondanks het feit dat in 1991 geen NTG-dagen zijn georganiseerd wordt het jaar afgesloten met een positief saldo. Dit positief saldo is voor het overgrote deel te danken aan een meevaller: het bleek dat NTG nog geld kreeg uit de eindafrekening van de NTG-dagen 1990. De verwachtte intering op het kapitaal heeft derhalve niet plaatsgevonden.

Bedragen in guldens	**Debet**	**Credit**
Contributie		10.625,00
Sponsoring		
Rente		249,82
Evenementen		1.589,75
Administratie	929,31	
KvK en notaris	61,00	
Bestuurskosten	460,40	
Computer faciliteiten		
Mededelingen a/d leden	6.717,70	
Reisbijdragen	1.991,05	
Representatie	174,00	
Diversen		
Saldo	2.131,11	
	12.464,57	12.464,57

Tabel 1: *De winst en verliesrekening over 1991*

2.1 Inkomsten

- **Contributies**
  De groei van het aantal leden is in 1991 doorgegegaan, waardoor aan het eind van 1991 143 (+26) geregistreerde en betalende leden aanwezig waren, waarvan 71 persoonlijke leden, twee studentleden en 26 instituutsleden met gezamenlijk 70 vertegenwoordigers. Dit betekent dat de contributie-inkomsten ongeveer *f* 1500,– hoger zijn dan begroot.
- **Rente**
  De inkomsten uit rente zijn lager dan begroot. Dit komt omdat de rente van de spaarrekening in het eerste kwartaal van het nieuwe jaar wordt uitgekeerd. Hiermee was bij het begroten geen rekening gehouden.
- **Evenementen**
  Hoewel dit jaar de NTG-dagen geen doorgang hebben kunnen vinden heeft NTG toch inkomsten uit evenementen gehad. Het grootste deel daarvan, *f* 1309,25 is afkomstig van de eindafrekening van de NTG-dagen 1990. De rest van het bedrag is overgebleven van de najaarsbijeenkomst en is bestemd om de rekening van de TUE te voldoen. Die rekening (van *f* 446,25) was op 31 december echter nog niet ontvangen. Het bedrag komt wél als verplichting op de balans naar voren.

2.2 Uitgaven

- **Administratie**
  Deze post bevat uitgaven voor het voeren van de administratie zoals portokosten, enveloppen, ordners. Het feit dat de vereniging groeit brengt hogere adminstratiekosten met zich mee. Ook de kosten van het verzenden van informatiemateriaal zijn in deze post opgenomen.
- **KvK en notaris**
  Deze post omvat de kosten van de inschrijving bij de kamer van koophandel in Groningen.
- **Bestuurskosten**
  Deze post bestaat uit de kosten van telefonische bestuursvergaderingen. De kosten blijken lager te zijn dan begroot was.
- **Mededelingen aan de leden**
  Deze post bevat de uitgaven gedaan voor het maken, reproduceren en verzenden van het ‘verslag met bijlage’. Ondanks het ruim hogere ledental zijn de kosten hiervan maar ongeveer 10% hoger dan verwacht.
- **Reisbijdragen**
  De voorzitter heeft in 1991 twee maal een reis gemaakt waarvoor hij een bijdrage van de NTG heeft gevraagd. De eerste reis was naar de TEX confe-

rentie in Dedham, VS. De tweede reis was naar de Europese TeX bijeenkomst in Parijs. Van beide bijeenkomsten is door hem verslag gedaan in de MAPS. Het budget dat voor dit doeleinde beschikbaar was is niet opgebruikt.

- **Representatie**
  Het bedrag dat bij deze post is opgenomen is uitgegeven aan attenties voor de organisatie van de NTG-dagen 1990 en het afgetreden bestuurslid. Ook de hotelkosten van de heer Taupin zijn hierin opgenomen. De totale uitgaven blijven ruim onder de begroting.
- **Saldo**
  Deze post is geen uitgave, maar het bedrag dat de vereniging in 1991 heeft overgehouden. Hiervan moet de eerder genoemde *f* 446,25 worden afgetrokken.

3 De balans

In tabel 2 is de balans per 1 januari 1991 weergegeven. daaruit blijkt dat in 1991 nog contributie voor het jaar 1990 geïnd moest worden.

Bedragen in guldens	**Aktiva**	**Passiva**
Giro	14.114,22	
Kas		
Contributies	200,00	
Cursusgeld		
Kapitaal		14.314,22
	14.314,22	14.314,22

Tabel 2: *De balans per 1 januari 1991*

In tabel 3 is de balans per 31 december 1991 weergegeven. Uit de balans blijkt dat de vereniging een financieel gezond jaar achter de rug heeft. Ook is te zien dat een aantal leden nog geen contributie heeft betaald. Degene(n) die op de ledenvergadering nog steeds niet aan hun verplichtingen over 1991 hebben voldaan zullen voor royement worden voorgedragen.

Daartegenover staat dat een aantal leden hun contributie voor het verenigingsjaar 1992 reeds vóór 31 december 1991 had betaald.

In 1992 wordt een cursus georganiseerd, twee deelnemers hebben de kosten reeds in 1991 voldaan.

Het al eerder genoemde bedrag van *f* 446,25 is een nog niet betaalde (op 31 december niet ontvangen) rekening voor de lunch tijdens de najaarsbijeenkomst.

Het NTG-kapitaal is met *f* 1684,86 toegenomen.

Bedragen in guldens	**Aktiva**	**Passiva**
Giro	19.703,23	
Kas	1,10	
Contributies	236,00	3.295,00
Cursusgeld		200,00
Verplichtingen		446,25
Kapitaal		15.999,08
	19.940,33	19.940,33

Tabel 3: *De balans per 31 december 1991*

4 Conclusie

De vereniging heeft andermaal een gezond jaar achter de rug. Dank zij een meevaller is een batig saldo het resultaat. Zonder die meevaller zou de vereniging ongeveer quite hebben gedraaid, hetgeen een positieve ontwikkeling is na de enigszins sombere begroting waarin een tekort werd verwacht.

NTG's listserver TEX-NL

22 april 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
VANELST@AMOLF.AMOLF.NL                    JAN VAN ELST
CI@ANALYSIS.RUG.AC.BE                     Chris Impens
mattes@AZU.INFORMATIK.UNI-STUTTGART.DE    Eberhard Mattes
raichle@AZU.INFORMATIK.UNI-STUTTGART.DE   Bernd Raichle
LMO@BGERUG51                              Benedict R. Verhegghe
SASKIA@BGUVM                              Saskia Beeser
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"
piet@CS.RUU.NL                            Piet van Oostrum
eijkhout@CS.UTK.EDU                       Victor Eijkhout
frankw@CWI.NL                             Frank van de Wiel
rvdh@CWI.NL                               Rob van der Horst
ernst@DCMR1.UUCP                          E.R. de Vreede
combee@DELGEO.NL                          leendert combee
X33@DHDURZ1                               Joachim Lammarsch
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
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
FRAMBACH@ECO.RUG.NL                       "Erik Frambach"
N.POPPELIER@ELSEVIER.NL                   "Nico Poppelier"
ludo@ET.KULEUVEN.AC.BE                    Vangilbergen Ludo
AJKRIJGSMAN@ET.TUDELFT.NL                 Ardjan Krijgsman
COMBEE@ET.TUDELFT.NL                      leendert combee
SPIT@EVALUN11                             "Werenfried Spit"
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
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
WBAHKUI@HDETUD1                           Gerard Kuiken
WIORA03@HDETUD1                           Netty Zuidervaart
DENHAAN@HDETUD5                           Jack den Haan
WBWEAHA@HDETUD51                          J.B.W. HOEBEEK
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
TAJ@HGRRUG5                               Theo Jurriens
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
SADARJOE@HLERUL53                         I.A. Sadarjoen
VDSCHOOT@HLERUL53                         Jan Vanderschoot
DAVID@HLERUL59                            David van Leeuwen
ZWIJNSVO@HLSDNL50                         C. VAN ZWIJNSVOORDE <ZWIJNSVOORDE@HLSD
MFAGKCHR@HMARL5                           CHRIS EVELO
U001290@HNYKUN11                          Niek Cox
U001310@HNYKUN11                          Ronald Kappert
U070040@HNYKUN11                          Patrick Wever
U212307@HNYKUN11                          Peter Bronts
U216002@HNYKUN11                          Paul Wackers
U250005@HNYKUN11                          Peter-Arno Coppen
U251006@HNYKUN11                          Hans Stoks
U253002@HNYKUN11                          Constant Cuypers
U279102@HNYKUN11                          Theo van den Heuvel
U439019@HNYKUN11                          Ton de Haan
U605005@HNYKUN11                          Willem Jan Karman
U605008@HNYKUN11                          Rik Fleuren
U641012@HNYKUN11                          Rini van Doorn
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
BARKEY@ITI.TNO.NL                         Chuck Barkey
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
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
aerts@MEDIA01.UUCP                        Ad H. Aerts
bob@MPI.KUN.NL                            Bob Boelhouwer
ian.nimmo-smith@MRC-APPLIED-PSYCHOLOGY.CAMBRIDGE.AC.UK  ian nimmo-smith
hdavids@MSWE.DNET.MS.PHILIPS.NL           Henk Davids
KOERDIEN@NLR.NL                           Koerdien van Wijk, NLR-NOP
kegel@NU.CS.FSU.EDU                       Harald Kegelmann
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
MOUCHE@RCL.WAU.NL                         Pierre von Mouche
WEITS@RCL.WAU.NL                          Ello Weits
J.L.Braams@RESEARCH.PTT.NL                Johannes L. Braams
egdnt01hbtex@RUG.NL                       Henk Brouwer
DALVAN@RUGR86.RUG.NL                      Rene van Dal
DINGS@RUGR86.RUG.NL                       Marcel Dings
nijhof@RUGTH4.TH.RUG.NL                   Jeroen Nijhof
rein@RUG4.CS.RUG.NL                       Rein Smedinga
koole@RULWI.LEIDENUNIV.NL                 Ger Koole
VDGRIEND@RULWINW.LEIDENUNIV.NL            J.A. van de Griend
ELBERS@SARA.NL                            Chris Elbers
petervc@SCI.KUN.NL                        Peter van Campen
lenssen@SG.TN.TUDELFT.NL                  K.-M. Lenssen
v912182@SI.HHS.NL                         Eric Veldhuyzen
tonnie@STACK.URC.TUE.NL                   Tonnie Geraets
POPPE@SWOV.NL                             "Frank Poppe"
SCHRAMA@TUDGV1.TUDELFT.NL                 Ernst J.O. Schrama
bax@TUDGW2.TUDELFT.NL                     Arjen Bax
HUISMAN@TUDW03.TUDELFT.NL                 H. Huisman
TH788310@TWNCTU01                         TSAI, YING-TEH
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
KNAPPEN@VKPMZD.KPH.UNI-MAINZ.DE           Joerg Knappen Uni-Mainz
KNAPPEN@VKPMZD.PHYSIK.UNI-MAINZ.DE        J"ORG KNAPPEN
dee@WLDELFT.NL                            Dick Dee
Marc.Kool@WLDELFT.NL                      Marc H. Kool
TAN@WLDELFT.NL                            "K H. Tan"
vdvoorn@WLDELFT.NL                        Marjan v\d Vooren
best@ZEUS.RIJNH.NL                        Robert W. Best
konings@ZEUS.RIJNH.NL                     Joop Konings
*
* Total number of "concealed" subscribers:         3
* Total number of users subscribed to the list:  167  (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.
- 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

22 april 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    39 91/02/13 10:49:00
  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  1307 92/02/06 23:11:17
  ARTIKEL1 STY        ALL NTG V      80   678 92/02/06 23:12:05
  ARTIKEL2 DOC        ALL NTG V      80  1264 92/02/06 23:13:50
  ARTIKEL2 STY        ALL NTG V      80   641 92/02/06 23:14:26
  ARTIKEL3 DOC        ALL NTG V      80  1341 92/02/06 23:16:11
  ARTIKEL3 STY        ALL NTG V      80   688 92/02/06 23:18:11
  RAPPORT1 DOC        ALL NTG V      80  1631 92/02/06 23:40:06
  RAPPORT1 STY        ALL NTG V      80   820 92/02/06 23:40:30
  RAPPORT2 DOC        ALL NTG .       .     0 ........ ........
  RAPPORT3 DOC        ALL NTG V      80  1629 92/02/06 23:18:48
  RAPPORT3 STY        ALL NTG V      80   812 92/02/06 23:28:40
  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 92/02/07 00:08:30
  NTGSTYLE UAB        ALL NTG V      80  1000 92/02/07 00:08:58
  NTGSTYLE UAC        ALL NTG V      80  1000 92/02/07 00:09:29
  NTGSTYLE UAD        ALL NTG V      80  1000 92/02/07 00:09:57
  NTGSTYLE UAE        ALL NTG V      80  1000 92/02/07 00:11:17
  NTGSTYLE UAF        ALL NTG V      80  1000 92/02/07 00:14:15
  NTGSTYLE UAG        ALL NTG V      80   488 92/02/07 00:15:11

*************************************************************************
*
*  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   469 92/03/12 00:49:27
  SUPERTAB STY        ALL NTG V      80   241 92/03/12 00:49:44
  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 UUE    : zoo.exe for MS-DOS, UUencoded
*  MSFIZ UUE    : fiz.exe for MS-DOS, UUencoded
*  MSSUP201 UUE : MS-DOS support for zoo, ZOO archive, UUencoded
*  Z201SRC1 UUE : Sources of zoo, part 1, ZOO archive, UUencoded
*  Z201SRC2 UUE : Sources of zoo, part 2, ZOO archive, UUencoded
*  TEXSHELL UUE : TeX environment for the Atari ST, ZOOed
*  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    UUE        ALL NTG V      61   946 90/03/18 20:53:06
  MSFIZ    UUE        ALL NTG V      61   297 90/03/18 20:54:11
  MSSUP201 UUE        ALL NTG V      61   621 90/03/18 20:55:29
  Z201SRC1 UUE        ALL NTG V      61  1975 90/03/18 20:58:36
  Z201SRC2 UUE        ALL NTG V      61  2046 90/03/18 20:59:21
  TEXSHELL UUE        ALL NTG V      62   594 90/01/03 16:12:11
  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

*************************************************************************
*
*    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`
- `rusinfo.rus.uni-stuttgart.de`
- `tex.ac.uk`
- `math.utah.edu`

of via e-mail bij:

- `mail-server@cs.ruu.nl`
- `mail-server@rusmv1.rus.uni-stuttgart.de`
- `LISTSERV@DHDURZ1`

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.

Working Group 1: Education

Review Michael Urban's 'An introduction to LATEX'

Kees van der Laan

February 1992

1 Compliments

To start with I like it, it is easy reading. A good style, no typos. Bravo! Because other people refer to this work and because TUG distributes it as number 9 in the TEXniques series—and therefore it might be considered as THE introduction to LATEX—it seemed still appropriate for me to review it. As always with introductions the challenge is to tell only 'white' lies when telling the incomplete story.[1] So my review will have the structure of enumerating—with annotations—what is treated in the syllabus and what I would consider essential. Of course the latter is a matter of taste. I shall adopt the author's point of view '...introduce you to the LATEX document preparation system ...' and discuss to what extend the author succeeded.

My comments and suggestions are about the February 86 version.

Some first impressions and overall remarks to start with. The syllabus does not contain exercises nor does it make use of simple diagrams to illustrate discussed items. An index is also missing.

2 Introduction

In the introduction the scope of the work '...introduce you to the LATEX document preparation system and get you working with LATEX as fast as possible' is stated. Disadvantages of word processor systems are enumerated

- not good at creating high-quality output with multiple fonts and sophisticated spacing,
- lack the power to do automatic sectioning, footnotes, tables of contents, cross-references and the like.[2]

Next the concept of style files is alluded to, and the power of LATEX summarized. It is also mentioned that simple documents are simple to prepare, in spite of the non-WYSIWYG way of working.

2.1 What I missed

I missed a discussion of the relation between (descriptive) mark-up and formatting, along with LATEX's place therein.

3 Getting started

The aim of this chapter: 'To produce your first LATEX document,' is easily reached. The 'General Operation' discussion could have benefitted from a simple flow diagram to accompany the process stages as explained in the text. The typing vs. typesetting paragraph treats: quotation marks, dashes, ligatures, and line breaking.

4 Control sequences

Enough is told about the escape character, the control sequences and the control symbols in order to use these for mark up.[3]

5 Environments

In the `\begin{...}` and `\end{...}` environments (by the way what a horrible typesetted title), the text elements are: quote, quotation, verse, verbatim, center, flushright, flushleft, and list.

5.1 What I missed

I missed how to create basic paragraph shapes. A additional remark about how to handle special paragraph shapes, or a reference to the literature for it, does not harm.

6 Putting it together

Page mark-up and descriptive mark-up are treated pêle-mêle. The logical structure of chapters, sections etc. and the front matter (title, table of contents) and back matter (index and in general appendices) aspects are

[1] And it is incomplete due to its conciseness: just 56 pages!

[2] Whether this is still the case or not I, the reviewer, consider LATEX's capability to work on document *parts* with the possibility to do referencing to other parts, very powerful.

[3] I consider it right at this level not to mention how to create your own commands. The reader at this level is probably more familiar with his smart editor in order to reduce retyping of document elements.

discussed. How to number the front (and/or back) matter separately from the main document is touched upon.

6.1 What I missed

The template in section 5.6 is nice, but how to process the document in parts should have been discussed. Furthermore, no footers nor running heads are treated. Also how to create marginal notes or footnotes is missing.

7 Optional: tables and math

The author provides optional chapters about table and math typesetting. This means that these chapters are loosely coupled to the previous ones and could be skipped.

In the tables chapter the concept of floating document elements is explained. The way how extra vertical white space can be created in order to paste in document elements prepared by other tools, is insufficiently treated. In this way open space can be split over page boundaries. Inevitably the introduction to the mark up of tabular material is elementary. I personally would prefer to mark up framed tables in two steps. First the table, and next—separately, as a general tool—the framing, which could have been applied to any document element. This is a general approach. One can think of encadring of words, letters or even quotations (to mix the mark up of the rules with the mark up of the table is the rule rather than the exception).

The treatment of how to mark up of mathematical material is understandably very elementary. For practical situations I consider this insufficient, especially with respect to aligned equations.[4]

7.1 What I missed

What I missed, especially in the math chapter, is a discussion of how math should appear in print, independently of the tool used. Mention Swanson at least. The (La)TeX way is taken for granted, but that is not enough due to the various ways formulas can be marked up.

8 Cross-referencing

Just enough is told in order to use symbolic referencing. It is not explained how to refer to items in the list of references.

9 Error messages

The last chapter deals with how to read, understand and cope with error messages; 7 pages! The author could not get around treating the box and the glue concepts, in order to explain the overfull and underfull messages. These concepts are fundamental however and should have been treated earlier, and not hidden in the error messages chapter.

10 Conclusion

It is pleasant reading—I found no typos! As a first introduction to the *use* of LaTeX—*as is*, by examples—it is good and well-written. However, after reading this syllabus an author is not capable of marking up his journal article or technical report via LaTeX. It is a *real* introduction. A list of (annotated) references for further reading would have served the reader. Exercises to accompany the first hands-on experience had to be added, along with their answers. Experience from the TeXbook has it that readers start from these answers as templates for similar situations.

References

[1] Urban, M. (1986): An introduction to LaTeX (to order from the TUG office).

[2] Swanson, E. (1986): Mathematics into Type, AMS.

[4] Personally, I don't like examples completely alienated from their original meaning. The example of δ_{ij} is offensive. Why had the example given at page 49 of Lamport's manual to be altered? If the Kronecker delta had to be introduced do it in the correct way.

Working Group 1: Education

Addendum 'Publiceren met LATEX'

Kees van der Laan

December 1991

1 Inleiding

De recensie van het boek 'Publiceren met LATEX' (CWI syllabus 19, 196 pp) in de Mededelingen van het Wiskundig Genootschap van Februari 91 is wat oppervlakkig. Het lijkt mij nuttig enige additionele kanttekeningen te plaatsen, te meer daar de syllabus geruime tijd geleden gemaakt is en bij cursussen gebruikt wordt. Ik ben van mening dat docenten en cursisten met deze opmerkingen hun voordeel kunnen doen. Het bijwerken van de syllabus is beduidend meer werk dan het geven van aanvullende opmerkingen. Of er ooit een bijgewerkte syllabus zal verschijnen is van vele factoren afhankelijk. Het wordt relevant als LATEX 3[1] verschenen is. Ondanks dat lxiii upwards compatible gedacht is, denk ik dat het bijwerken van de syllabus dan opportuun zal zijn, tenzij er andere introducties beschikbaar zijn. Op het ogenblik is dit niet het geval in het Nederlands. De introducties in andere talen die ik ken hebben allen een ander uitgangspunt: het introduceren van LATEX. Een Nederlandse introductie is relevant m.b.t. de Europese conventies van pagina formaat en opmaak, de afbreekpatronen, en de reserved words. Zie de rapportages van de NTG werkgroep Neerlandica hierover, o.a. in MAPS 90.2. en Braams (1991).

Het uitgangspunt van de 'Publiceren met LATEX' syllabus: LATEX voor het maken van publicaties gebruiken—*as is*—is prima en daar sta ik nog steeds helemaal achter.[2]

2 Begrippen

In het begrippen hoofdstuk zou aandacht besteed moeten worden aan SGML.[3] Daarnaast moet ingegaan worden op zaken zoals genoemd in Southall (1984).

3 Eenvoudig LATEX

Uitgegaan had moeten worden van hoe-het-behoort, los van de te gebruiken techniek. Bijvoorbeeld van Treebus (1988), of internationaler The Chicago Manual of Style.

Het voorbeeld in paragraaf 2.6.1 is niet correct: `\bf` opdrachten zijn weggevallen.

Bij de paragraaf over invoertekens, 2.3, had aangegeven moeten worden hoe de backslash zelf gezet kan worden in de tekst en in voetnoten e.d. In het algemeen moet aangegeven worden hoe in titels, voetnoten e.d. verbatim tekst opgenomen kan worden.

Het gebruik van `\section*{...}` versus het globale aanduiden van het onderdrukken van nummering via `\setcounter...` moet besproken worden.

Naast de LATEX standaard stijlen zijn er nu diverse andere stijlen beschikbaar, o.a. de Nederlandse stijlen, en internationaler AMS-LATEX, en TUGboat's LTUGboat.sty.[4]

Ingegaan had kunnen worden op de basis structuren van paragrafen zoals behandeld door Wittbecker (1988).[5] Ook zou een verwijzing gegeven kunnen worden over het creëren van endnotes.[6]

[1] Zoals bekend werken Frank Mittelbach, Rainer Schöpf, Chris Rowley, Johannes Braams en anderen aan een nieuwe versie van LATEX, het zogenaamde LATEX 3 project—voor intimi lxiii—opgestart in Stanford89. Optimistische schattingen noemen 1993 als het jaar waarin LATEX 3 opgeleverd wordt.

[2] Een tipje van de sluier. Momenteel werk ik aan: Publishing with TEX, volgens dezelfde opzet, en nog wat meer.

[3] Zie o.a Van Herwijnen's Practical SGML, 1990. Voor de relatie SGML-TEX is van der Laan (1990b) relevant. Voor de situatie bij een uitgever—Elsevier Science Publishers—zie Poppelier (1990).

[4] Als men aan een boek werkt is het raadzaam zo snel mogelijk afspraken te maken met de uitgever over de invoer en opmaak van de kopij.

[5] Voor speciale paragraafstructuren zie bijvoorbeeld Eijkhout (1990a), en ermee gerelateerd Eijkhout (1990b,c).

[6] Basismateriaal daarvoor vormt Durst (1990) en Salomon (1990 m.n. p. 591). Maar dat is misschien niet realistisch gezien de moeilijkheidsgraad van deze tutorials. Bovendien gaat dit beduidend verder dan—*as is*—gebruiken, en is kennis van plain TEX nodig.

Aangegeven had moeten worden hoe in een enumerate omgeving de items alfabetisch genummerd kunnen worden.

Het aanduiden dat publicaties over ‘spelletjes’ met LATEX geformatterd kunnen worden, kan nu beduidend beter dan het opnemen van een voorbeeld uit de bridge typografie. Verwijzingen naar publicaties (en macros) over Bridge (van der Laan, 1989 en 1990a), Cross words (Brian Hamilton Kelly, 1990; van der Laan, 1992; Go (Kołodziejska, 1991), en schaken (o.a. Tutelaers, 1991) zouden opgenomen kunnen worden.

Aan taal-specifieke eigenschappen moet aandacht besteed worden, m.n. aan de Babel-optie, Braams (1991), voor het verkrijgen van ‘reserved words’ in de eigen taal, zoals hoofdstuk, literatuur e.d. Ook voor andere specifieke Nederlandstalige aspecten, zoals de invoer van woorden met een trema, het zetten van aanhalingstekens openen, en het gebruik van de ij-ligatuur, is Babel relevant. Daarnaast zou op meertalige documenten ingegaan kunnen worden, en verwezen kunnen worden naar Ferguson’s werk, m.n. naar INRS-TEX, en zijn multilingual reporten, o.a. in TUGboat. Ook kan het geen kwaad te noemen dat niet-Westerse schriftsoorten een heel eigen problematiek kennen.

In verband met tekstverwerkers zou conversie programmatuur genoemd kunnen worden, onder vermelding dat de huidige stand van zaken nog veel te wensen overlaat, zie van der Laan & Luyten (1988). Overwogen moet worden dat kopij opgemaakt met een word processor veelal eenvoudiger geTEXed kan worden uitgaande van de ASCII uitvoer van de word processor.

Het invoeren van kopij via Beebe’s LATEX intelligente EMACS editor, of via Williams & Hall (1990) intelligente EDT, moeten genoemd worden. Ook kan verwezen worden naar de publicaties over Wordperfect als editor voor (La)TEX, zie Hoover (1989), Modest (1989). Daarnaast is er zoiets als La-check, dat de invoer checkt op incorrectheid, m.n. of er geen gekromde haakjes ontbreken (in Beebe’s editor wordt dit voorkomen omdat de editor de omgeving al voortypt, en er *binnen* de omgeving slechts *in*getypt hoeft te worden).

4 Wiskundige formules

Dit hoofdstuk moet grondig gereviseerd worden. Het bevat wiskundige onjuistheden. Er moet uitgegaan worden van hoe wiskunde gezet behoort te worden, onafhankelijk van de formatting taal. Swanson (1986), is daarvoor een goede bron.

Verder moet op zijn minst verwezen worden naar AMS-LATEX voor stijlen en extra fonts.

Voor de revisie kan geput worden uit mijn Math into BLUes (1991) en in het algemeen uit Mittelbach (1989b).

De uitwerkingen van de opgaven moeten beter. De dubbele punt, als relationeel symbool, moet anders ingevoerd worden dan de dubbele punt als leesteken. In het algemeen heeft dit te maken met formula classes. Dit hele begrip ontbreekt, helaas.

De afsluitende punt in de uitwerking van opgave 9 is foutief geplaatst, zie TEXbook, 18.1 Punctuation. De uitwerking van het commutator diagram is misplaatst en bovendien niet correct (het zetten van pijlen van ongelijke lengte had ook mooi geïllustreerd kunnen worden). Voor commutator diagrammen kan het best verwezen worden naar Spivak’s LaMSTEX. Het is in plain TEX maar mogelijk te gebruiken binnen LATEX. Borceux (1990) gebruikt LATEX’s picture omgeving en is voor LATEX gebruikers misschien het meest relevant.

5 Tabellen

Ingegaan had moeten worden op ‘landscape’ vs. ‘portret’ afdrukken, ook al kan dit veelal via een driver optie gerealiseerd worden. Ook had conform de inleiding ingegaan moeten worden op tabellen die zich uitstrekken over meerdere paginas, m.n. Jurriens’ supertabular.sty had genoemd moeten worden. Het werk van Cowan, waarbij de template automatisch bepaald wordt uit de tabel inhoud, moet genoemd worden. Het vermelden van Khanh Ha’s Easy table voor het maken van fraaie en complexe tabelkoppen via eenvoudige opdrachten kan geen kwaad (dat Mittelbach de tabular omgeving implementatie heeft verbeterd kan achter de schermen blijven).

In paragraaf 4.3 e.v. moet de uitlijning op de decimale punt fraaier, o.a. het onderdrukken van de decimale punt wanneer deze niet nodig is.

Een omlijnde tabel kan beter en universeler verkregen worden via een `\framebox` om een tabel (ook het LATEX manual zet gebruikers hieromtrent op het verkeerde been).

In de uitwerkingen van de opgaven hadden teksten in de kop van de tabel verticaal gecentreerd kunnen worden. Bij opgave 2 is het hinderlijk dat een essentieel 2-koloms tabel als een 4-koloms tabel gebracht wordt.

In het ‘invulformulier’ had gebruik gemaakt moeten worden van ‘leaders’ en/of van betere (verticale) witruimten.

Bij de tabel uit het spoorboekje had het verschijnsel van voetnoten bij een tabel en het gebruik van speciale tekens toegelicht kunnen worden.

6 Illustraties

In het hoofdstuk over illustraties is Figuur 5.1 nodeloos ingewikkeld.

Het hoofdstuk heeft als ruggegraat LATEX’s beperking dat slechts lijnen onder een beperkt aantal richtingen gezet kunnen worden. Jammer. De bezier.sty optie had genoemd kunnen worden waarmee zelfs taartdiagrammen gemaakt kunnen worden. Bezier.sty vraagt veel geheugen maar dat is niet erg omdat figuren apart gezet kunnen worden, en er open ruimtes in de tekst gereserveerd kunnen worden. Spivak heeft extra 'line-fonts' vrij beschikbaar bij zijn LaMSTEX, dat onafhankelijk ervan te gebruiken is. Hiermee zijn dus lijnen onder meerdere richtingen te verkrijgen. Voor ad hoc schuine lijntjes zou ook gebruik gemaakt kunnen worden van Hendrickson (1985).

Ingegaan had moeten worden op het maken van een grafiek uitgaande van een file van gegevens, zie Jurriens' 'From observation to publication'. Hij gebruikt PiCTEX.

In het algemeen moet ingegaan worden op het 'electronisch plakken' van documentdelen via encapsulated Postscript, zie Anita Hoover's compilatie van de huidige werkwijzen hieromtrent in de proceedings van TUG91 (of MAPS 91.2).

Verwijzingen naar de literatuur i.v.m. chemische formules (o.a. Haas & O'Kane; 1989)[7], staafdiagrammen (o.a. Nagy (1989) en Schöpf (1990)), taartdiagrammen, bomen (Eppstein (1985), Brüggeman-Klein (1989)) c.q. grafen zou het geheel danig completeren. De mogelijkheden van gnu-plot, met de muis de tekening maken op het scherm, en gnu-plot levert de LATEX opdrachten, maakt het invoeren gemakkelijker (ik heb er geen ervaring mee en weet niet hoe geschikt het is t.a.v. wijzigingen c.q. parameters).

7 Fonts

In het hoofdstuk over fonts had gebruikt gemaakt kunnen worden van Southall (1985), i.v.m. de te gebruiken terminologie.

METAFONT op zich moet genoemd worden, bijvoorbeeld i.v.m. de eenvoudige mogelijkheid om schrift om te keren. De creatie van een nieuw font gaat veel te ver, maar op zich is het vermelden van het werk van Velthuis een goede zaak. Ook zou ingegaan kunnen worden op schaak fonts en het go font (zie MAPS 90.2).

Het meest indrukwekkende is het vermelden van het zetten van muziek, hetzij via de Poolse school (METAFONT nodig) hetzij via Taupin's werk, MAPS 92.1, waarbij ook speciale symbolen via METAFONT gecreeerd zijn.

Verwijzing naar bronnen voor andere fonts is op zijn plaats, o.a. naar Quin (1990).

Bovendien zou nu ingegaan moeten worden op het hanteren van encapsulated Postscript.

Aparte vermelding is nodig van virtuele fonts, m.n. het nut ervan bij het plaatsen van 'ongewone' accenten, zoals bijvoorbeeld in het Tsjechisch.[8] Font families en het font selection schema hadden behandeld moeten worden (Mittelbach & Schöpf; 1990).

Het gebruik van Sowa's BM2fonts voor het converteren van bitmap Logo's is het noemen waard en zou mooi geïllustreerd kunnen worden aan de hand van het RUG-wapen.[9]

Ook zou verwezen kunnen worden naar Hoenig's werk over het zetten van tekst langs gekromde lijnen via interactie met METAFONT.[10]

8 Van hoofdstukken naar rapport

Er kan nu mooi aangesloten worden bij Smedinga's ervaring: Hoe met LATEX een boek gemaakt kan worden, 1991 (MAPS 91.2).

Het voorbeeld met het maatlatje vind ik misleidend, ook al is het uit het LATEX manual overgenomen, p. 197, figure c.6. Er wordt verborgen gehouden dat LATEX counters met de waarde 0 niet afdrukt!

Het begrip `\changebars` en hoe deze te realiseren, eventueel via een verwijzing, zou op zijn plaats geweest zijn.[11]

Hoe men ongenummerde items in een toc-file (table-of-contents) kan opnemen had genoemd moeten worden. Ook had het toegepast moeten worden bij de productie van het boekje zelf: het is hinderlijk dat 'Literatuur' met paginanummer ontbreekt in de inhoudsopgave, te meer daar de literatuurlijst zich niet aan het eind bevindt. Iets minder hinderlijk is het ontbreken van de verwijzing naar 'Inleiding' in de inhoudsopgave.

Het maken van een index is stiefmoederlijk bedeeld en behoeft aandacht.

9 Nog wat te wensen?

Informatie over NTG (Nederlandstalige TEX Gebruikersgroep) en TUG (de internationale TEX Users Group), moet opgenomen worden. In ieder geval moet dit niet verborgen worden in een voorbeeld over een brief (de uitnodiging voor de oprichtingsbijeenkomst van de NTG) c.q. in een voorbeeld over meerkolomsuitvoer met namen van de Board of Directors van TUG.

In het algemeen moet er meer aandacht besteed worden aan de verschillende mogelijkheden van het 'vloeien ' van tekst om figuren (Kneser, 1990; en in Salomon's tutorials). De eerste letter van een hoofdstuk als miniatuur is een voorbeeld daarvan (via de zogenaamde drop macro).

Aan het maken van informatica publicaties m.n. het zetten van programma teksten zou aandacht besteed

[7] Zie ook het verslag van het KNCV symposium 'Molecuul Muis Manuscript' in MAPS 92.1.

[8] Zie Zlatuska (1991).

[9] Zie Sowa (1991).

[10] Zie Hoenig (1991).

[11] Zie eventueel Salomon's eerder genoemde tutorials.

moeten worden (Van Oostrum, 1991).

Ook is een appendix met Hoenig's TEX for new users, op zijn plaats omdat het een overzicht geeft van het gehele TEX gebeuren. In dat kader kan ook preventief ingegaan worden op de 'frequently asked questions.'

Voor systeembeheerders is Schrod's overzicht van de componenten mogelijk nuttig. Dit geldt idem dito voor Mulders' TEX structuurschemas. Het noemen van Cheswick's index kan geen kwaad ook al is de creatie van reeds bestaande LATEX opdrachten beschermd.

Natuurlijk moet ook de literatuurlijst bijgewerkt worden. Ook moeten andere introducties tot LATEX genoemd worden zoals

- Urban, M (1986): An introduction to LATEX,
- Bürger, D (1990): LATEX for scientists and engineers. MC Graw-Hill,
- Kopka, H (1989): LATEX, Eine Einführung. Addison-Wesley.

Voor een lijst van LATEX boeken zie de resource guide van TUG of raadpleeg TUGlib. Voor degenen die zelf stylen ontwikkelen is het goed van het bestaan te weten van de `doc-` en `autodoc-styles` van Mittelbach en Hamilton Kelly. Overigens zijn de mogelijkheden van Vector-TEX indrukwekkend.

10 Conclusie

Het boekje is goedkoop en bevat uitgewerkte voorbeelden. De meeste van de bovengenoemde aspecten betreffen nieuwere ontwikkelingen, vandaar Addenda. Te overwegen is het boekje te reviseren voor de huidige doelgroep. Voor de geavanceerden zou er een ander boekje moeten verschijnen over geavanceerd gebruik en style ontwikkeling. De Nederlandstalige markt is daarvoor te klein, en het zou het beste in het Engels kunnen. Hopelijk gebeurt dat als zij-effect van het lxiii project.

References

[1] Alexander, J.C (1986): Tib, a reference setting package. TUGboat, 7, 3, 138–139. (An update note is in TUGboat, 8, 2, 102.).

[2] AMS-LATEX. AMS.

[3] Appelt, W (1988): Typesetting Chess. TUGboat, 9, 3, 284–287.

[4] Beebe, N.H.F (1992): LATEX Editing Support. Te verschijnen in TUGboat en MAPS 92.1.

[5] Bezier.sty (op de file servers).

[6] Bodenheimer, B (1990): Frequently asked questions. comp.text.tex newsgroup. Adapted by Van Oostrum in MAPS 91.1, 85–92.

[7] Borceux, F (1990): De la construction de diagrammes. Cahiers GUTenberg, 5, 41–48.

[8] Braams, J.L (1991): Babel, a multilingual style-option system for use with LATEX's standard document styles. TUGboat, 12, 2, 291–301, met een voorloper in MAPS 91.1.

[9] Brüggemann-Klein, A & D. Wood (1989): Drawing trees nicely with TEX. EP-ODD, 2, 2, 101–115.

[10] Bruin, R. de, C.G van der Laan, J.R. Luyten, H.F. Vogt (1988): Publiceren met LATEX. CWI Syllabus 19.

[11] Cheswick, B (1990): A permuted index for TEX and LATEX. CSR 145. AT&T (ook in de TEXniques reeks).

[12] The Chicago Manual of Style (1982). University of Chicago Press.

[13] Cowan, R.F (1985): Making tables with macros. tables.tex & tabledoc.tex (op de file servers).

[14] Durst, L.K (1990): Long-winded endnotes and exercises with hints or solutions. TUGboat, 11, 4, 580–588 (a comment is given in Hefferon (1991): Getting `\answers` in TEX).

[15] Durst, L.K (1991): Some tools for making indexes: Part I. TUGboat, 12, 2, 248–258.

[16] Eijkhout, V (1990a): Unusual paragraph shapes. TUGBoat, 11, 1, 51–53.

[17] Eijkhout, V (1990b: An indentation scheme. TUGBoat, 11, 4, 613–616 (ook in MAPS 90.2).

[18] Eijkhout, V (1990c): A `\parskip` scheme. TUGBoat, 11, 1, 616–619 (ook in MAPS 90.2).

[19] Eppstein, D (1985): Trees in TEX. TUGboat, 6, 1, 31–37.

[20] Ferguson, M.J (1991): INRS-TEX (new version). Zie ook zijn multilingual reporten, o.a. in TUGboat, 11, 4, 514–515.

[21] Geest, L. van & M. van Geest (1991): Gebruik van TEX en LATEX op het CAWCS. MAPS 91.2, 48–56 (bevat o.a. macros voor Nassi Schneidermann diagrammen in LATEX. Macros op de file servers).

[22] Haas, R.T. & O'Kane, K.C. (1987):Typesetting Chemical Structure Formulas with the Text Formatter TEX/LATEX. Comput. Chem, 11.4, 251–271.

[23] Hamilton Kelly, B (1989): The autodoc-option. TUGboat, 10, 2, 274–284.

[24] Hamilton Kelly, B (1990): Some macros to draw crosswords. TUGboat, 11, 1, 103–119.

[25] Hendrickson, A (1985): Some diagonal line hacks. TUGboat, 6, 2. 83–86 (the macros go wrong for (nearly) vertical lines. Consult Amy for a more recent version).

[26] Herwijnen, E. van (1990): Practical SGML. Kluwer.

[27] Hoenig, A.J (1991a): Typesetting along arbitrary curves with TEX and METAFONT. TUG91 proceedings. TUGboat, 12, 4, 554–557.

[28] Hoenig, A.J (1991b): TEX for new users. O.a. MAPS 91.2, 91–96.

[29] Hoover, A.Z (1989): Using wordperfect 5.0 to create TEX and LATEX documents. TUGboat, 10, 4, 549–559.

[30] Hoover, A.Z (1991): Getting Postscript into TEX and LATEX documents. TUG91 proceedings (met overdruk in MAPS 91.2).

[31] Johnson, D.B (1990): Changebar document style option (op de fileservers).

[32] Jurriens, T. A (priv. comm.): Supertabular.sty (op de file servers).

[33] Jurriens, T. A (1992): From observation to publication, MAPS 92.1.

[34] Khanh, Ha (1990): Easy Table. TUGboat, 11, 2, 250–264.

[35] Kneser, T (1990): LaTeX-paragraphs floating around figures. TUGboat, 12, 1, 28–30.

[36] Kolodziejska, H (1991): Go diagrams with TeX. MAPS 91.2, 63–68 (op file servers).

[37] Laan, C.G. van der, J.R. Luyten (1988): Evaluation of K-talk. RC-RUG rapport 15 (bevat o.a. test collectie voor evaluatie van converters).

[38] Laan, C.G. van der (1989): Typesetting Bridge via LaTeX. TUGboat, 10, 1, 113–116 (met een kopie in MAPS 91.2).

[39] Laan, C.G. van der (1990a): Typesetting Bridge via TeX. TUGboat, 11, 2, 265–276 (met een voorloper in GUTenberg cahiers 5, en een kopie in MAPS 91.2).

[40] Laan, C.G. van der (1990b): SGML(, TeX and ...). TUGboat, 12, 1, 90–104. (Met een voorloper in MAPS 90.2 en GUTenberg cahiers 5.). Het bevat bovendien een uitgebreide bibliografie. SGML-TeX referenties zijn daarom achterwege gelaten in de lijst hier.

[41] Laan, C.G. van der (1991): Math into BLUes (1991). TUG91 proceedings voor deel I en het dubbelnummer GUTenberg Cahiers 10&11 voor deel II, met een voorloper in MAPS 91.1.

[42] Laan, C.G. van der (1992a): Typesetting Crosswords via TeX. MAPS 92.1.

[43] Laan, C.G. van der (1992b): Table Diversions. MAPS 92.2. (Contribution for EuroTeX92.)

[44] Larson, J.E (1989): A chess font for TeX. TUGboat, 10, 3, 351.

[45] McClure, M (1989): TeX macros for COBOL syntax diagrams. TUGboat, 10, 4, 743–750.

[46] Mittelbach, F (1988): A new implementation of the array and tabular environments. TUGboat, 9, 3, 298–313. (Correction TUGboat, 10, 1, 103-104.)

[47] Mittelbach, F (1989a): The doc-option. TUGboat, 10, 2, 245–273.

[48] Mittelbach, F (1989b): An environment for multicolumn output. TUGboat, 10, 3, 407–415.

[49] Mittelbach, F (1989c): An extension for the LaTeX theorem environment. TUGboat, 10, 3, 416–426.

[50] Mittelbach, F & R. Schöpf (1989): A new font selection scheme for TeX macro packages—the basic macros. TUGboat, 10, 2, 222–273.

[51] Mittelbach, F & R. Schöpf (1990a): The new font selection scheme—User interface to standard LaTeX. TUGboat, 11, 2, 297–305. (Corrected version of TUGboat 11, 1.)

[52] Mittelbach, F & R. Schöpf (1990b): Towards LaTeX 3.0. TUGboat, 12, 1, 74–79. (Ook in MAPS 90.2. Status rapport MAPS 92.1.)

[53] Modest, M.F (1989): Using TeX and LaTeX with Wordperfect5.0. TUGboat, 10, 1, 67–72.

[54] Mulders, H.P.A (1990): TeX structuurschemas. MAPS 90.2. 109–112.

[55] Nagy Dezső (1989): A bar chart in LaTeX. TUGboat, 10, 2, 239–240.

[56] Oostrum, P. van (198?): Fancy headings. (Op de file servers.)

[57] Oostrum, P. van (1991): Program text generation with TeX/LaTeX. MAPS 91.1, 99–105. (Een overzicht van de beschikbare gereedschappen.)

[58] Poppelier, N.A.F.M (1990): SGML and TeX in scientific publishing. TUGboat, 12, 1, 105–109. (Met een kopie in MAPS 90.2.).

[59] Quin, L.R.E (1990): Summary of METAFONT fonts available. TeXMaG, 4, 6, 1990 (Overdruk in MAPS 91.1).

[60] Read, D (priv.comm.): Some ideas to improve LaTeX. (Rapport met enige suggesties m.b.t. de LaTeX implementatie, geïnspireerd op SGML.)

[61] Rubinstein, Z (1989): Chess printing via METAFONT and TeX. TUGboat, 10, 2, 170–172. (Met erratum in TUGboat, 10, 3, 359.).

[62] Rubinstein, Z (1989): Printing annotated chess literature in natural notation. TUGboat, 10, 3, 387–390.

[63] Salomon, D (1989): Macros for indexing and table of contents preparation. TUGboat, 10, 3, 394–400. (Commentaar door Breitenlohner (1990): TUGboat, 11, 1, 62, m.b.t. het schrijven van lange records naar `\write` streams. Zie MAPS 92.1 voor nieuwere versie.)

[64] Salomon, D (1991): Output Routines Examples & Techniques.
Part I: Introduction & Examples. TUGboat, 11, 1, 69–85.
Part II: OTR Techniques. TUGboat, 11, 2, 212–236.
Part III: Insertions. TUGboat, 11, 4, 588–605.

[65] Sankar, S (1989): APE—a set of TeX macros to format ADA programs. TUGboat, 10, 1, 89–97.

[66] Schöpf, R (1989): Drawing histogram bars inside the LaTeX picture-environment. TUGboat, 10, 1, 105–108.

[67] Schrod, J (1991): The components of TeX. MAPS 92.1. (Eerdere versies in Die TeXnische Komöde, en TeXline 14.)

[68] Smedinga, R (1991): Hoe met LaTeX een boek gemaakt kan worden. MAPS 91.2, 97–101.

[69] Southall, R (1984): First principles of typographic design for document production. TUGboat, 5, 2, 79–90, 1984. (Corrigenda (1985): TUGboat, 6, 1, p.6).

[70] Southall, R (1985): Designing a new typeface via METAFONT. STAN-CS-85-1074, 1985.

[71] Sowa, F (1991): Graphics and halftones with BM2font. TUG91 proceedings. TUGboat 12, 4, 534–538.

[72] Spivak, M (1989): LaMSTeX. TeXplorators Corporation.

[73] Swanson, E (1986): Mathematics into Type, AMS.

[74] Swonk, G.L (1991): LaTeX tree drawer. TUGboat, 12, 2, 286–289.

[75] Taupin, D (1990): Musique en TeX. Cahiers GUTenberg, 5, 62–69. Engelse (en vernieuwde) versie in MAPS 92.1.

[76] Tofsted, D (1990): An improved chess font. TUGboat, 11, 4, 542–544.

[77] Tutelaers, P (1991): A font and a style for typesetting chess using LaTeX or TeX. MAPS 91.2, 41–45. (Submitted to TUGboat.).

[78] Treebus, K (1988): Tekstwijzer, een gids voor het grafisch verwerken van tekst. SDU. Den Haag.

[79] Whitney, R & B.N. Beeton (1989): TUGboat authors’ guide. TUGboat, 10, 3, 378–385. (Voor gebruik van o.a. LTUGboat.sty.)

[80] Wichura, M.J (199?): The PiCTeX manual. TeXniques, 6. (≈$30,–.)

[81] Williams, L & L. Hall (1990): Increased efficiency using advanced EDT editing features. TUGboat, 11, 3, 421–424.

[82] Williams, T & C. Kelley (199?): GNUPLOT, an interactive plotting program. (Van de file servers.).

[83] Wittbecker, A (1988): Making paragraphs. TUGboat, 9, 3, 272–276.

[84] Zlatuska, J (1991): Virtual fonts with accented letters. Cahiers GUTenberg 10&11, 57–68.

Werkgroep 4: Fonts

Met schuine en begerige ogen. ‘Peremesjtsjenije’.

Een verhaal van nette armoei

Erik-Jan Vens

Computer Modern? Acht, het blijft een crime. De letter gaat kapot zodra-die op minder dan 1000dpi afgedrukt wordt. En toch is het de met TeX meegeleverde letter, dus iedereen gebruikt ’em. De enige andere METAFONT letter in het publieke domein is de Pandora. Dit is een letter die het uitstekend doet op lage resoluties. En ja, . . ., (etc.) je wilt toch wel eens wat anders.

En met begerige ogen kijk je naar wat voor lettertypes de PostScript gebruikers tot hun beschikking hebben. Maar ja, dat zijn rijke stinkers. Die hebben hun dure PostScript printers & die kopen rustig een Times Roman setje voor ƒ 795. En de arme sloeber wil ook zo’n grote keuzevrijheid hebben, en z’n Trabantje inruilen voor een Škoda, en z’n Škoda inruilen voor een Audi 100. De arme sloeber wil kunnen ftp-en naar `cica.cica.indiana.edu` en de publiek domeine Garamond kunnen downloaden. En behalve downloaden ook echt gebruiken. Wat zou hij[1] dan tevreden zijn. Helaas, pindakaas, maar wie voor een dubbeltje geboren is, wordt nooit een kwartje.

Tenzij je je toevlucht zoekt tot list en bedrog. Waar twee honden vechten om één been, gaat een derde er mee heen. De firma Adobe was zo vriendelijk (?, ja, vriendelijk) het formaat van hun Type One fonts te publiceren. Maar dat scheelt een hoop! Want nu kan iedereen met een beetje inzicht in programmeren

```
10 PRINT ERIK-JAN IS GEK
20 GOTO 10
```

zo’n PostScript Type One lettertype uit elkaar slopen en naar de TeX wereld vertalen. De één denkt dan meteen aan de puntjes waaruit een letter voor de printer is opgebouwd, de ander aan de vectorbeschrijving, maar je ziet alle butjes[2] denken, piekeren, kraken.

Marcel Dings begon een vertaler te schrijven van PostScript Type One naar METAFONT. Dat bleek een haalbare kaart. Maar zijn programma & de tijd die hij er aan kon besteden bleken weinig uitbreidbaar. Daar ik vroeger ook wel eens een lesje programmeren gehad heb, besloot ik er mijn visie eens tegen aan te bemoeien. En omdat ik een soepele baas heb, die mij veel speelruimte biedt, is er een werkbaar product uitgekomen.

Ook de arme TeX-freak kun nu het Internet af op zoek naar vertaalbare lettertypes. Wat dat oplevert zullen we in het volgende voorbeeld tegenkomen. Wanneer we de publieke domein versie van een Baskerville uit de voor mensen onleesbare binaire representatie naar leesbare PostScript vertalen, kunnen we het volgende tegenkomen:

```
/A { -3 723 hsbw -3 230 vstem 452 275
   vstem 226 42 hstem 1 20 hstem 659
   20 hstem 385 678 rmoveto -37 hlineto
   -240 -581 rlineto -22 -53 -29 -12
   -57 -1 rrcurveto -31 vlineto 230
   hlineto 31 vlineto -21 hlineto -27
   -35 4 35 hvcurveto 0 9 3 9 4 9
   rrcurveto 52 129 rlineto 266 hlineto
   54 -129 rlineto 5 -11 4 -12 0 -12
   rrcurveto -29 -37 -2 -21 vhcurveto
   -22 hlineto -31 vlineto 275 hlineto
   31 vlineto -49 1 -25 21 -19 44
   rrcurveto -252 581 rlineto closepath
   68 -410 rmoveto -231 hlineto 108 267
   rlineto 3 8 2 8 3 9 rrcurveto 7 -27
   11 -27 10 -26 rrcurveto 87 -212
   rlineto closepath endchar } ND
```

Zodra je beseft dat er een omgekeerd Poolse notatie wordt gebruikt –dus niet “`hstem 20 1`”, maar “`1 20 hstem`”– en dat alle ‘keywords’ een betekenis hebben (zo betekent “-22 hlineto”: trek vanuit het huidige punt een lijn ter lengte van 22 eenheden naar links) voel je op je klompen aan dat er een eenduidige afbeelding in METAFONT te vinden is. Voor het voorbeeld zou dat iets worden als:

```
   (x0, y0) -- (x0 - 22, y0)
```

De METAFONT-implementatie wordt iets ingewikkelder wanneer je alle “`hstem`” en “`vstem`” opdrachten wilt meenemen, want dat zijn resolutie-afhankelijke opdrachten. Maar dat is op te lossen. Bovendien gebruikt PostScript een andere fontindeling dan TeX, maar ook dat is op te lossen.

In METAFONT zou de bovenbeschreven letter er alsvolgt uitzien:

[1] Nee, daar heeft U geen gelijk in, het is een hij.

[2] Dat is een woord dat ik geleerd heb toen ik in Groningen kwam wonen & dat ik onvertaald laat.

```
beginchar(65,723*FX#,678*FY#,0*FY#);
"A";
ah((382,678)
lt(345,678)
lt(105,97)
ct(83,44,54,32,-3,31)
lt(-3,0)
lt(227,0)
lt(227,31)
lt(206,31)
ct(179,31,144,35,144,70)
ct(144,79,147,88,151,97)
lt(203,226)
lt(469,226)
lt(523,97)
ct(528,86,532,74,532,62)
ct(532,33,495,31,474,31)
lt(452,31)
lt(452,0)
lt(727,0)
lt(727,31)
ct(678,32,653,53,634,97)
lt(382,678))cp
ah((450,268)
lt(219,268)
lt(327,535)
ct(330,543,332,551,335,560)
ct(342,533,353,506,363,480)
lt(450,268))cp
chp[65]:=currentpicture;
endchar;
```

(Voor de insider, ik heb de hint-informatie even verwijderd.) In dit stukje METAFONT-code staan een aantal macros, zoals `lt`. Dit staat voor “lineto” en is geïmplementeerd als:

```
def lt (expr a,b) = -- (a,b) enddef;
```

Redelijk rechttoe-rechtaan.

Wanneer je deze letter afdrukt, ziet-ie er alsvolgt uit:

A

Kortom, de wereld van de arme sloeber heeft zich weer eens verbreed. Niet alleen hoeft-ie zich nu zorgen te maken over z’n tekst & over de gebruikte layout, maar hij kan zich nu ook zorgen maken over de te kiezen letter.

Wie eens een versie van dit pakket wil uitproberen –het is nog niet officieel uit– kan contact opnemen met:

```
Erik-Jan Vens
RuG/ICCE
Postbus 335
9700 AV  Groningen
(0 50) 63 36 49
```

Of met Email een briefje sturen naar:
`E.J.Vens@icce.rug.nl`.

Werkgroep7: PC-zaken

Jos Winnink

winnink@ecn.nl

Algemeen

Wegens gebrek aan informatie over andere pc-platformen wordt er in dit stukje alleen aandacht besteed aan MS/PC-DOS systemen.

Mocht er informatie zijn over andere pc-platformen (Atari, Amiga, Acorn, Apple Macintosh ...) die ook voor andere gebruikers nuttig is dan hierbij het verzoek om deze informatie aan de redactie van de MAPS te sturen (liefst in electronische vorm), zodat ook andere gebruikers van deze informatie kunnen profiteren.

MS-DOS

In de vorige MAPS werd meegedeeld dat emTEX zich wat betreft de versie 3.14 van TEX in een β-testfase bevindt. Inmiddels geldt dit voor de implementatie van versie 3.141. Bij voortduring worden met name in de BIGTEX-versie en de 386-versie verbeteringen aangebracht.

Van meer belang zijn volgens mij de wijzigingen in de programma's die de DVI file's verwerken. Het resultaat is dat mede door gebruik van lange namen voor opties (als alternatief voor korte *command line* opties) in combinatie *respons-files* het verwerken van DVI-files aanzienlijk is vereenvoudigd. Deze vereenvoudigingen bewijzen zich ondermeer wanneer men meerdere pagina's op 1 bladzijde wil plaatsen (ook bij het previewen werkt dit).

Ook is het mogelijk om indien dit gewenst is ontbrekende *fonts* naar behoefte te genereren.

Door de organisatie van de bijgeleverde DVI-driver's is er nagenoeg geen apparaat dat niet met emTEX kan worden aangestuurd. Voor POSTSCRIPT kan gebruikt worden gemaakt van DVIPS.

De kwaliteit van dit produkt uit zich ook in het feit dat regelmatig op de emTEX discussielijst vragen verschijnen van mensen uit alle delen van de wereld. Nog steeds wordt de distributie ook in Nederland met enige regelmaat naar belangstellenden verstuurd.

Dit brengt mij op de NTG-TEX distributie voor pc's. Moest de vorige keer nog gemeld worden dat het zoeken was naar een spellingcontrole programma inmiddels is dat probleem ook opgelost omdat via Erik Frambach een dergelijk programma beschikbaar kwam dat geschreven is door A. Merckens. Hoewel dit programma zich nog in een fase bevond dat met name de documentatie nog niet volledig gereed was lijkt het aan de behoefte te kunnen voldoen.

Al met al zij we nu in een situatie aangeland dat de distributie in elkaar kan worden gezet. Hopelijk zijn deze werkzaamheden bij de eerstvolgende NTG-bijeenkomst van juni 1992 afgerond.

MusicTEX
Using TEX to write polyphonic or instrumental music

Daniel Taupin

Laboratoire de Physique des Solides
(associé au CNRS)
bâtiment 510, Centre Universitaire,
F-91405 ORSAY Cedex

1 What is MusicTEX ?

MusicTEX is a set of TEX macros to typeset polyphonic, orchestral or polyphonic music.

Two *sizes* are available : 16pt and 20pt (standard) staff heights. For that purpose, it uses special *fonts* : `musicn16`, `slurn16`, `beamn16` and `musicn20`, `slurn20`, `beamn20` respectively.

It is to be emphasized that MusicTEX is not intended to be a compiler which would translate into TEX some standard musical notations, nor to decide by itself about aesthetic problems in music typing. MusicTEX only typesets staves, notes, chords, beams, slurs and ornaments as requested by the composer. Since it makes very few typesetting decisions, MusicTEX appears to be a versatile and rather powerful tool. However, due to the important amount of informations to be provided to the typesetting process, coding MusicTEX might appear to be awfully complicated, just as the real keyboard or orchestral music. It should be interfaced therefore by some pre-compiler in the case of the composer/typesetter wanting aesthetic decisions to be automatically made by somebody (or something) else.

1.1 Music typesetting is two-dimensional

Most of the people who just learnt a bit of music at college probably think that music is a linear sequence of symbols, just as litterary texts to be TEX-ed. In fact, with the exception of strictly monodic instruments like most orchestral wind instruments, one should be aware that reading music actually is a matricial operation : the musician successively reads *columns* of simultaneous notes. This is the reason why, in **MusicTEX** the fundamental *macro* is of the form

```
\notes ... & ... & ... \enotes
```

where the character & is used to separate the notes to be typeset on respective staffs of the various instruments, starting from the bottom.

In the case of an instrument whose score has to be written with several staffs, these staffs are separated by the character `|`. Thus, a score written for a keyboard instrument and a monodic instrument (for example piano and violin) will be coded as follows :

```
\notes ... | ... & ... \enotes
```

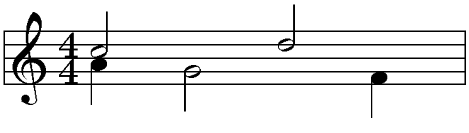
for each column of simultaneous notes.

1.2 The spacing of the notes

It seems that many books have dealt with this problem. Although it can lead to interesting algorithms, we think it is a rather minor one.

In fact, each column of notes has not necessarily the same spacing and, in principle, this *spacing* should depend on the shortest duration of the simultaneous notes. But this cannot be established as a rule, for at least two reasons :

1. spacing does not depend only of the local notes, but also on the context, at least in the same bar.
2. in the case of polyphonic music, exceptions can easily be found. Here is an example :



where it can be clearly seen that the half notes at beats 2 and 3 must be spaced as if they were quarter notes, since they overlap, which is obvious only because of the presence of the indication of the *meter* 4/4.

Therefore, we prefered to provide the composer/typesetter with a set of macros, the spacing of which increases by a factor of $\sqrt{2}$ (incidentally, this can be adjusted) :

```
\notes ... & ... & ... \enotes  % 1 spatial unit
\Notes ... & ... & ... \enotes  % 1.4 spacial unit
\NOtes ... & ... & ... \enotes  % 2 spatial units
\NOTes ... & ... & ... \enotes  % 2.8 spatial units
\NOTEs ... & ... & ... \enotes  % 4 spatial units
\NOTES ... & ... & ... \enotes  % 5.6 spatial units
```

The size of the spatial unit (`\elemskip`) can be freely adjusted. In addition, MusicTEX provides a means of adjusting the note spacing according to an average number of elementary spaces within a line (macro `\autolines`).

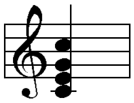
1.3 Music tokens, rather than a readymade generator

The tokens provided by MusicTEX are :

- the note symbols *without stem* ;
- the note symbols *with stems, and hooks for eighth notes and beyond* ;
- the indications of beam beginnings and beam ends ;
- the indications of beginnings and ends of ties and slurs ;
- the indications of accidentals ;
- the ornaments : arpeggios, trills, mordents, pincés, turns, staccatos and pizzicatos, fermatas ;
- the bars, the meter and signature changes, etc.

Thus, `\wh g` produces an *A (445 Hz)* whose duration is a *whole note*. In the same way, `\qu c` produces a *C (250 Hz approx.)* whose value is a *quarter note with stem up*, `\cl J` produces a *C (125 Hz approx.)* whose duration is an *eighth note with stem down*, etc.

To generate quarter, eighth, sixteenth, etc. chords, the macro `\zq` can be used : it produces a quarter note head whose position is memorized and recalled when another stemmed note (possibly with a hook) is coded ; then the stem is adjusted to link all simultaneous notes. Thus, the perfect C-major chord, i.e.

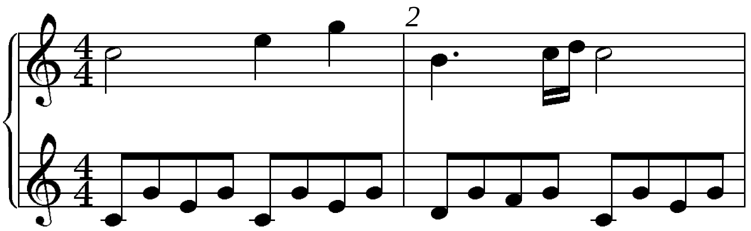


is coded `\zq c\zq e\zq g\qu j` or, in a more concise way, `\zq{ceg}\qu j` (stem up) : in fact, single notes are treated. . . like one-note chords.

Beams are generated using macros which define their beginning (at the current horizontal position), together with their altitude, their sense (upper of lower), their multiplicity, their slope and their reference number. This latter feature – the reference number – appears to be necessary, since one may want to write beams whose horizontal extents overlap : therefore, it is necessary to specify which beam the notes hang on and which beam is terminated at a given position.

Besides, a general macro (`\zcharnote`) provides a means of putting any sequence of symbols (in fact, some `\hbox{...}`) at any pitch of any staff of any instrument. Thus, any symbol defined in a font (letters, math symbols, etc.) can be used to typeset music.

Before entering details, we give below an example of the two first bars of the sonata in C-major by MOZART (Mozart, K545) :



The *coding* is set as follows :

```
\def\nbinstruments{1}\relax    % a single instrument
\nbporteesi=2\relax            % with two staffs
\generalmeter{\meterfrac{4}{4}}\relax    % meter is 4/4
\debutmorceau          % initiates staffs, clefs, indication of the meter.
\normal                % normal spacing (14pt for \Notes)
\temps\Notes\ibu0f0\qh0{cge}\tbu0\qh0g|\hl j\enotes
    % \ibu0f0  begins an upper beam, aligned on the f, reference number 0, slope 0.
    % \tbu0  terminates this beam before writing the second g by means of \qh0g,
    % where \qh..indicates a note hanging on a beam.
\temps\Notes\ibu0f0\qh0{cge}\tbu0\qh0g|\ql l\sk\ql n\enotes
    % \sk   sets a space between the two quarters at the right hand, so that
    % the second is aligned with the third eighth of the left hand.
```

```
\barre    % bar.
\Notes\ibu0f0\qh0{dgf}|\qlp i\enotes    % \qlp = quarter with a point.
\notes\tbu0\qh0g|\ibbl1j3\qb1j\tbl1\qb1k\enotes
   % \ibbl1j3 begins a double beam, aligned on the C (j at this pitch) of slope 0.15.
\temps\Notes\ibu0f0\qh0{cge}\tbu0\qh0g|\hl j\enotes
\suspmorceau    % closing with a simple bar (for short excerpts).
```

1.4 Other highlights

MusicTEX is intended to handle up to six instruments, each of which may be typeset on zero to four staffs (zero staff is used to record the text of vocal parts).

Signatures are stated either for all instruments, such as : `\generalsignature=-2` which sets two flats on each staff, but this can be partly overrided by `\signatureii=1` which puts one sharp on the staffs of *instrument number* 2 (ii). Of course, signature may change at any time as well as meters and clefs.

Besides, two frequently posed questions deserve an answer. The first one is : *"can MusicTEX transpose a score ?"*. The answer is now 99.5 % *yes*. If fact, there is an internal register named `\transpose` the default value of which is zero, but it may be set to any positive of negative reasonable value. Then, it offsets all symbols pitched with letter symbols by that number of pitch steps. However, it will not change the signature, and if – for example – you transpose by 1 pitch a piece written in C, it will not know whether you want it in $D\flat$, in D or in $D\sharp$. This might become tricky if accidentals occur within the piece, which might have to be converted into flats, naturals, sharps or double sharps, depending on the new choosen signature. To circumvent this trouble, *relative* accidentals have been implemented, the actual output of which depends of the pitch of this accidental and of the current signature.

The second question is : *"can I write an orchestral score and extract the separate scores for individual instruments ?"* The answer is 95 % *yes* : in fact, you can define your own macros `\mynotes...\enotes`, `\myNotes...\enotes` with as many arguments as there are in the orchestral score (hope this is less or equal to 9, but TEXperts know how to work around) and change its definition depending on the selected instrument (or insert a test on the value of some selection register). But the limitation is that the numbering of instruments may change, so that `\signatureiii` may have to become `\signaturei` if instrument iii is alone. But this is not a serious problem for average TEX wizard apprentices.

1.5 How to get it

The whole *distribution* fits into a single 1.2Mbyte or 1.44Mbyte diskette. It can also be obtained through an *anonymous ftp* at `rsovax.ups.circe.fr` (130.84.128.100), after selecting the subdirectory `[.musictex]`. All sources (including fonts) are provided, either separately, or "zipped" or as VMS 'savesets".

1.6 Implementation and restrictions

The macroinstruction file MusicTEX contains approximately 2500 lines of code, that is 80 000 bytes approximately. This requires your score to be compiled by the most extended versions of TEX (65 000 words of working memory). In desperate situations, we recommend using the "BigTEX" processors which, unfortunately, perform a great deal of disk input/outputs (on PCs) which make them awfully slow.

In particular, the number of registers it uses makes it doubtfully compatible with LaTEX.

Ties and slurs have been implemented in a way which may look rather ugly, but we think it is the only way of implementing *in one pass* ties and slurs which run *across glue*. The principle is to have tie/slur symbols with a rather long part of horizontal stuff. Then, at each time a glue occurs and at each time a group of notes is coded while a slur or tie is pending, an `\hrule` is issued which overlaps the preceeding tie/slur symbol so that the final output seems to contain a continuous line. Unfortunately, this is possible only in the glue expansion direction, namely in the horizontal direction.

1.7 Acknowledgements

The idea of implementing the present package is due to the previous work (MUTEX) of Andrea STEINBACH and Angelika SCHOFER[1]. This work provided the basis of the Metafont codes and some line breaking procedures, which both are still used here... with 99% corrections and updates.

2 Practical use

2.1 Heading statements

Before any reference to MusicTEX macros :

```
\input musicnft
\input musictex
```

which may be followed by `\input musicadd` in the case you have more than six instruments (voice is two instruments: one for the music, one for the text) or more than 6 simultaneous beams or ties or slurs.

[1] Steinbach A. & Schofer A., *Theses* (1987, 1988), Rheinische Friedrich-Wilhelms Universität, Bonn, Germany.

After that, you may write a complete book of TEX provided that you do not use & as a tabulation character (its `\catcode` has been changed) and that you do not overwrite MusicTEX's definitions. This means that no special macros have been designed to help you write titles, author names, comments, literature excerpts, etc.

2.2 Before you begin to write notes

You should first specify whether you want to typeset music in size 20pt par staff or 16pt. This only optional, the default value being 20pt. If you want the 16pt size, then you have to say :

```
\musicsize=16
```

Then, the first compulsory declaration is :

```
\def\nbinstruments{n}
```

where n is the number of instruments, used by MusicTEX to performs loops building staffs, setting signatures, meters, etc. Therefore, it must be defined before any other statements. An instrument may consist of several staffs, e.g. the piano. The difference between one instrument of several staffs and several instruments is as follows :

- distinct instruments may have distinct *signatures*, distinct staffs of a unique instrument share the same signature.
- *stems* may be hung to *beams* belonging to differents staffs of the same instrument.
- *chords* may extend across several staffs of the same instrument.
- staffs of a unique *instrument* are tied together with a big brace at the beginning of each line.

If the number of staffs (in french "*portées*") is not equal to one, this number must be specified by :

```
\nbporteer=p\relax
```

where p is the number of staffs, and where r is the roman numeral of the instrument considered (e.g. `\nbporteesiii` for the 3rd insrument, starting from the bottom).

If the signature is not void, one should code :

```
\generalsignature=s\relax
```

where $s > 0$ indicates the number of *sharps* in the signature and $s < 0$ the number of *flats*[2].

If there a *meter* indication is to be posted, it should be specified using the macro

```
\generalmeter{m}%
```

where m is the description of the meter indication which should appear on each staff. If it is a *fraction* (e.g. 3/4) on should code

```
\generalmeter{\meterfrac{3}{4}}%
```

or, in a simpler way (if the numbers are less than 10) :

```
\generalmeter{\meterfrac 34}%
```

Special denotations can be used, such as `\allabreve` to get 𝄵 and `\meterC` to get 𝄴.

However, not all music scores have the same meter in each staff. Especially, some staffs may have *ternary* meters while others have *binary*. This can be specified by using the `\generalmeter` macro to set the meter for most of the scores and overriding it by means of a more sophisticated command :

```
\metertoksii={{\meterfrac{12}8}{%
\allabreve}{}{}}%
```

which sets the meter to 12/8 for the first (lower) staff, and *alla breve* for the second staff of the instrument number 2 (*ii*). Note that there is room for 4 staffs and that void items must be specified, otherwise TEX weird errors occur.

If you want the *name of the instrument*s (or the *name of the voice*s) to be displayed in front of their respective staffs at the beginning, you may code :

```
\def\instrumentr{name of the instrument}%
```

where r is the roman numeral of the instrument considered. In this case, you should also adjust the `\parindent` dimension so that the long name of an instrument does not spill too far into the left margin.

2.3 Starting your masterpiece

Just code

```
\debutmorceau
```

which will initiate (with indentation `\parindent`) the first set of staffs for all instruments you have previously defined. But that is not sufficient to begin writing notes and silences. In fact, you also must choose the spacing of the notes. This is done by saying

```
\normal
```

if you want a elementary spacing of `10pt`. If you say `\large` the elementary spacing (`\elemskip`) is set to `12pt`. Once this is done, you can select $\sqrt{2}$ multiples of this spacing by initiating your column with `\notes` (spacing `\elemskip`), `\Notes` (spacing `1.4\elemskip`), etc. If these values of `\elemskip` are not fit to your needs, you can also say

```
\normal\elemskip=15pt
```

Of course, if your are sufficiently skilled with TEX, you can also compute `\elemskip` according to the number of bars you want in a line and the number of notes in each bar. Incidentally, this is done by the `\autolines` macro which will be introduced later.

[2] We have seen once a score in G-minor where the signature consisted in two flats (B and E) plus one sharp (F). This is not supported by MusicTEX.

In practice, the choice of the macro `\notes`, `\Notes`, `\NOtes`, etc., to initiate of column of notes sets an internal dimension register, named `\noteskip` to the given multiple of `\elemskip`. The text of the macros `\normal`, `\large`[3] and `\etroit` is thus rather simple :

```
\def\normal{\ifdim\Interligne<0.1pt%
   \computewidths\fi % (ignore this safety)
\elemskip=2\Interligne\relax %
            % (i.e. 10 points)
\def\notes{\vnotes 1.0\elemskip }%
\def\Notes{\vnotes 1.4\elemskip }%
\def\NOtes{\vnotes 2.0\elemskip }%
\def\NOTes{\vnotes 2.8\elemskip }%
\def\NOTEs{\vnotes 4.0\elemskip }%
\def\NOTES{\vnotes 5.6\elemskip }}%

\def\large{\normal\elemskip=2.4%
\Interligne}%

\def\etroit{\ifdim\Interligne<0.1pt%
\computewidths\fi
\elemskip=2\Interligne\relax
\def\notes{\vnotes 1.0\elemskip }%
\def\Notes{\vnotes 1.3\elemskip }%
\def\NOtes{\vnotes 1.8\elemskip }%
\def\NOTes{\vnotes 2.6\elemskip }%
\def\NOTEs{\vnotes 3.6\elemskip }%
\def\NOTES{\vnotes 5.2\elemskip }}%

\def\vnotes#1\elemskip{\noteskip=#1%
\elemskip\n@otes}%
```

As seen above, if these standard spacings do not meet your requirements, you are welcome to define other spacings, using the more basic macro `\vnotes`.

However, you may also prefer to ask MusicTEX to compute the size of `\elemskip` from the page width (`\hsize`) after assuming a constant number of bars of a constant number of beats within each line[4]. In this case, you have better use the `\autolines` macro, which is described below (see : "Line and page breaking").

2.4 Note pitch specification

Note pitches are usually specified by letters ranging from `a` to `z` for those which are usually written under the G-clef (`a` corresponds to the A of frequency 222.5 Hz ; the *G* of the G-clef is denoted `g`). Lower pitch notes are specified using uppercase letters ranging from `A` to `N` (the *F* of the F-clef is denoted `M`, and `F` is one octave below).

If necessary, a numeric symbol can be used to place a symbol independently of the active clef. Besides, notes below `A` (i.e. the A of frequency 55.625 Hz), namely the lowest octave of the modern pianos can only be coded using the transposition features (see below : *transposition* and *octaviation*) or in absolute vertical position using numbers.

2.5 Writing notes

There are two major kinds of note macros :

1. those which terminate a note/chord stem and cause horizontal spacing,
2. those which initiate or extend a note/chord stem and do not cause horizontal spacing.

The first kind is used to type a melody, the second kind is used to type chords.

2.5.1 Single (spacing) notes

`\wh` p : whole note at pitch p.

`\hu` p : half note at pitch p with stem up.

`\hl` p : half note at pitch p with stem down.

`\qu` p : quarter note at pitch p with stem up.

`\ql` p : quarter note at pitch p with stem down.

`\cu` p : eighth note[5] at pitch p with stem up.

`\cl` p : eighth note at pitch p with stem down.

`\ccu` p : sixteenth note at pitch p with stem up.

`\ccl` p : sixteenth note at pitch p with stem down.

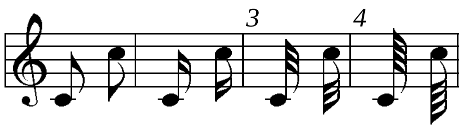
`\cccu` p : 32-th note at pitch p with stem up.

`\cccl` p : 32-th note at pitch p with stem down.

`\ccccu` p : 64-th note at pitch p with stem up.

`\ccccl` p : 64-th note at pitch p with stem down.

As an example, the sequence :



was coded as :

```
\notes\cu c\temps\cl j\enotes\barre
\notes\ccu c\temps\ccl j\enotes\barre
\notes\cccu c\temps\cccl j\enotes\barre
\notes\ccccu c\temps\ccccl j\enotes
```

If these notes are preceeded by *non-spacing* notes (i.e. macros `\zq` or `\zh`) their stem is extended up or down so as to join all notes into a single chord.

[3] To avoid problems with the LaTEX macro of the same name, this macros is only activated under LaTEX when `\begin{music}` is invoked.

[4] This does not meet the requirements of contemporaneous music, but fits very well to baroque and romantic music.

[5] The `\c` of this macro name is taken from the french word "croche" which is by the way one half of the english "crotchet" ; `\cc...`, `\ccc...` are standing for "double croche", "triple croche", etc.

2.5.2 Non-spacing (chord) notes

`\zq` p **:** quarter (or shorter) note head at pitch p with no spacing after.

`\zh` p **:** half note head at pitch p with no spacing after.

It must be pointed out that the pitch p of these notes is memorized so that the stem of the further spacing note will join them into a chord. This stem top and bottom pitch is *reset* at each spacing note.

REMARK : Notes of duration longer than whole notes are always non-spacing. This saves one useless definition, since this notes are always longer than other simultaneous ones. If needed they can be followed by `\sk` *to force spacing.*

2.5.3 Shifted non-spacing (chord) heads

These symbols are used mainly in chords where second intervals are present. It is the responsibility of the typist to choose which heads should be shifted left or right.

`\rw` p : whole note head shifted right by one note width (ca. 6pt), no spacing.
`\lw` p : whole note head shifted left by one note width (ca. 6pt), no spacing.
`\rh` p : half note head shifted right by one note width (ca. 6pt), no spacing.
`\lh` p : half note head shifted left by one note width (ca. 6pt), no spacing.
`\rq` p : quarter note head shifted right by one note width (ca. 6pt), no spacing.
`\lq` p : quarter note head shifted left by one note width (ca. 6pt), no spacing.

Except that they are shifted left of right, these macros act like `\z...` macros for stem building.

2.5.4 Shifted notes

`\rqu`**:** acts like `\qu` but the note head is shifted one note width. This is used for *chords* whose upper note has to be on the right side of the stem.

`\rhu`**:** same as above for a half note.

2.5.5 Non-spacing single notes

`\zhu`**:** half note with stem up but no spacing. It acts like `\hu` for stem building.

`\zhl`**:** half note with stem down but no spacing. It acts like `\hl` for stem building.

`\zqu`**:** quarter note with stem up but no spacing. It acts like `\qu` for stem building.

`\zql`**:** quarter note with stem down but no spacing. It acts like `\ql` for stem building.

`\lhu`, `\lhl`, `\lqu`, `\lql`**:** same as above, but the whole of the note is shifted one note width on the left.

`\zw` p **:** whole note at pitch p with no spacing after.

`\zwq` p **:** arbitrary duration note (𝅜) at pitch p with no spacing after.

`\zbv` p **:** breve note (▬) at pitch p with no spacing after.

`\zsb` p **:** semi-breve note (▭) at pitch p with no spacing after.

2.5.6 Pointed notes

One simple way of doing consists in putting `\pt` p to get a *dot* after the normal notehead at pitch p. Thus a quarter note with a point can be coded `\pt h\qu h`.

A simpler way of doing consists in using compact macros, namely : `\qup`, `\qupp`, `\quppp`, `\zqp`, `\zhp`, `\zwp` (these three `\z...p` are useful in chords), `\hup`, `\whp`, `\qhp`, `\qhpp`, `\qlp`, `\qlpp`, etc.

You may also introduce pointed notes, especially in groups by coding : `\qu{.a.^b.c}` which is equivalent to :

```
\pt{a}\qu{a}\pt{b}\sh{b}\qu{b}%
\pt{c}\qu{c}
```

2.6 Beams

Beams are not automatically handled, but they must be declared explicitely, *before* the first spacing note involving them is coded. Two kinds of macros are provided :

1. fixed slope beams have an arbitrary slope choosen by the user in the range -45% to +45% (by multiples of 5%) ;
2. semi-automatic beams have their slope computed knowing the number of `\noteskip` over which they are supposed to extend, and knowing the initial and final pitch of the notes they are supposed to link.

2.6.1 Fixed slope beams

`\ibu` nps **:** initiates an *upper beam* 3 horizontal line spacings above the pitch p. m is its reference number, which must be in the range [0-6] ([0-9] if `musicadd` file has been `\input`).

s is the slope of the beam. It is an integer in the range [-9,9]. $s = 1$ means a slope of 5%, $s = 9$ means a slope of 45% (the maximum with the `beamn20` or `beamn16` fonts), $s = -3$ means a slope of -15%, etc. With usual spacings a slope of 2 or 3 is fit for ascending scales. A slope of 6 to 9 is fit for ascending arpeggios.

`\ibl` nps **:** initiates a *lower beam* 3 horizontal line spacings below the pitch p. Other parameters as above.

`\ibbu` nps **:** initiates a *double upper beam* (same parameter meaning).

`\ibbl` nps **:** initiates a *double lower beam* (same parameter meaning).

`\ibbbu` nps **:** initiates a *triple upper beam* (same parameter meaning).

`\ibbbl` nps **:** initiates a *triple lower beam* (same parameter meaning).

`\ibbbbu` nps **:** initiates a *quadruple upper beam* (same parameter meaning).

`\ibbbbl` nps **:** initiates a *quadruple lower beam* (same parameter meaning).

Beam termination is also not automatic. The termination of a given beam must be explicitely declared *before* coding the last spacing note connected to that beam.

`\tbu` n **:** terminates upper beam number n at current position.

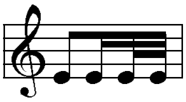
`\tbl` n **:** terminates lower beam number n at current position.

`\tbu` and `\tbl` terminate beams of any multiplicity. Therefore 32-th notes hanging on a triple beam are initiated by `\ibbbu` nps and terminated by `\tbu` n.

It is also possible to code beams whose multiplicity is not the same at the beginning. The multiplicity can be increased at any position. For instance, `\nbbu` n which sets the muliplicity of upper beam number n to 2 starting at the current position, `\nbbbu` n sets its multiciplicity to 3, and `\nbbbbu` n sets it to 4. `\nbbl` n . . . `\nbbbbl` n perform the same functions for lower beams.

Notes hanging or standing on beams are coded in the form `\qh`n p and `\qb`n p where n is the beam number and p the pitch of the note head. MusicTEX adjusts the length of the note stem to link the bottom of the chord to an upper beam (normally with `\qh`) and the top of the chord to a lower beam (normally with `\qb`).

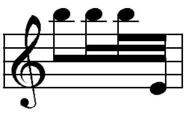
Note that the difference between upper and lower beams does not mainly consist in the beam being above or below the note heads ; rather, it specifies whether the abscissa of the beginning and the end of this beam is aligned on the right (upper beam) or on the left (lower) beam. Thus, the sequence :



has been coded as

```
\notes\ibu0e0\qh0e\nbbu0\qh0e\nbbbu0%
             \qh0e\tbu0\qh0e\enotes
```

It is quite possible to terminate with `\tbu` a beam initiated with `\ibl`. This may give :

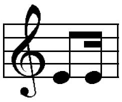


which has been coded as

```
\notes\ibl0p0\qb0p\nbbl0\qb0p\nbbbl0%
             \qb0p\tbu0\qh0e\enotes
```

Partial termination of beams is also possible, by using `\tbbu` or `\tbbl` : these macros terminate the current beam except that of order 1 (eights). `\tbbbu` or `\tbbbl` terminate the current beam except those of order 1 and 2, etc.

The macros `\tbbu` and `\tbbl` may also be invoked when only a single beam is active. Then, a second beam (upper or lower according the initiating procedure) is opened *one note width before the current position, and closed immediately*. Thus the following sequence

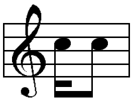


is coded :

```
\notes\ibu0e0\qh0e\tbbu0%
\tbu0\qh0e\enotes
```

The same behaviour occurs in the case of `\tbbbu`, `\tbbbl`, `\tbbbbu` and `\tbbbbl`.

The symmetrical pattern is also possible. For example :



has been coded as :

```
\Notes\ibbl0j0\rlap{\qsk\tbbl0}%
\qb0j\tbl0\qb0j\enotes
```

REMARK : these codings may seem complicated. In fact, it is the responsability of the user to define macros which perform the most common sequences in his masterpiece. For example, one could define sets of four sixteenths by the macro :

```
\def\qqh#1#2#3#4#5{%
\ibbl0#2#1\qh #2\qh #3\tbl0\qh #4}
```

where the first argument is the slope and the other four arguments are the pitches of the four consecutive sixteenths wanted.

2.6.2 Semi-automatic beams

In order to avoid tedious checks to adjust the slope (and even the starting pitch) of beams in music with a lot of steep beams, a set of automatically slope computing has recently been implemented (file `musicvbm.tex`). If you say `\Ibu2gj3` MusicTEX will understand that you want to build an upper beam (beam number 2) horizontally extending 3`\noteskip` and whose first note is a `g` and whose last note is `aj`. Knowing these parameters it will choose the highest slope number which corresponds to a slope not more than (`j` − `g`)/(3`\noteskip`). Moreover, if there is no sufficiently steep beam slope available, then it will raise the starting point.

Eight such macros are available : `\Ibu`, `\Ibbu`, `\Ibbbu`, `\Ibbbbu`, `\Ibl`, `\Ibbl`, `\Ibbbl` and `\Ibbbbl`. Examples of their use is given in `marcello.tex`.

2.7 Phantom notes

It may be interesting, when coding a sequence of notes within a unique pair `\notes...\enotes`, to skip one note place in order – for example – to set the third note of one staff at the same abscissa as that of the second note of another staff. This can be done by inserting `\sk` which causes a spacing of one `\noteskip`[6].

If you just want to shift a note or a symbol by one note head width, you may write `\qsk`.

Note that these two latter macros must be used inside a pair `\notes...\enotes`. If you want to make a spacing of one note head width oustside, write `\nspace`.

2.8 Collective coding : sequences of notes

As seen in the MOZART example, it is not necessary to write a macro sequence `\notes...\enotes` for each column. If, on all staffs of all instruments, spacings are equal or multiple of a unique value, the notes may be concatenated in each staff : each note in each staff makes the current position horizontally advance by the elementary spacing specified by the choice of `\notes`, `\Notes`, `\NOtes`, etc.

The major interest of this feature resides in that fact that the note macros are able to write several items ; for instance `\qu{cdefghij}` writes the *C-major* scale in quarters with stem up. In the same way `\cl{abcdef^gh}` writes the *A-minor* scale in eighths. Not all note generating macros can be used to perform collective coding, but most of them can. They are :

- all the spacing notes : `\wh`, `\hu`, `\hl`... `\ccccl` and the beam hooked notes, i.e. `\qh`n and `\qb`n.
- all the chord notes whose name is of the form `\z...`

Conversely, shifted notes are not supposed to be used in collective coding, mainly because they are used in very special cases which do not deserve wasting memory space to make them collective.

If necessary a void space can be inserted in a collective coding by using `*`.

2.9 Accidentals

Accidentals can be introduced in two ways.

The first way, the *manual* way of coding them, consists for example in coding `\fl a` to put a *flat* at the pitch a, supposedly before the further note of that pitch. There is no control upon the fact that a note will be put at this position and at this pitch. Naturals, sharps, double flats and double sharps are coded `\na` p, `\sh` p, `\dfl` p and `\dsh` p respectively.

Alternate procedures `\lfl`, `\lna`, `\lsh`, `\ldfl` and `\ldsh` place the same accidentals, but their abscissa is shifted one note head width on the left. The purpose of this is to avoid collision of accidentals in a chord with narrow intervals.

The second way of coding accidentals consists in putting the symbol ^ (sharp), the symbol _ (flat), the symbol = (natural), the symbol > (double sharp), or the symbol < (double flat) within the coding of the note, e.g.: `\qh{^g}` yields a $G\sharp$. This may very well be combined with collective coding, e.g.: `\qu{ac^d}`.

Two sizes are available for accidentals. They revert to the small version when notes are supposed to be too close to each other. These two sizes can be forces by coding `\bigfl`, `\bigsh`, etc., or `\smallfl`, `\smallsh`, etc. If one does not want to have any small accidentals, then one can declare `\bigaccid` (conversely `\smallaccid` or `\varaccid` – the latter restoring variable sizes).

It is also possible to put (small) accidentals *above* the note heads. This is done using `\uppersh` p, `\upperna` p or `\upperfl` p.

2.10 Transposition and octaviation

An important feature is the existence of a special register `\transpose` whose normal value if 0. If you say

```
\transpose=3
```

all subsequent pitches specified by upper or lower case letters will be transposed 3 positions. If you set `\transpose` to 7 you may write your music one octave below its final pitch. Thus, you can define *octaviation* macros like

```
\def\soqu#1{\zq{#1}{\transpose=7\relax
\qu{#1}}}
```

to build quarternote octaves in a single call. Note that the octaviated note is coded within braces so that the transposition is only local.

Octaviation can also be performed in is another way, namely unsing special codes to transpose by multiples of 7 intervals. For example `\qu{'ab}` is equivalent to `\qu{hi}` and `\qu{`kl}` is equivalent to `\qu{de}`. It should be emphasized here that the `'` (*acute accent*) and the `` ` `` (*grave accent*) have cumulative ef-

[6] Do not use `\kern` nor `\hskip` : in fact `\sk` not only causes a space but also records that space for correct handling of beams.

fects, so that `\qu{''A'A}` is equivalent to `\qu{ah}` and that the `\transpose` parameter is only reset to its initial value (not necessarily zero) when changing staff or instrument (i.e. `|` or `&`) or at `\enotes`. Since this may be confusing, it is useful tu use the `!` prefix to reset the `\transpose` register explicitely to the value it had when entering `\notes`[7]. Thus `\qu{!a'a}` always gives the note `a` and its upper octave `h` *shifted by the value of* `\transpose` *at the beginning of the current* `\notes...\enotes` group (or `\Notes...\enotes`, etc.) whatever the number of previous grave and acute accents occurring inbetween.

The above processes indeed change the vertical position of the note heads and associated symbols (note stems, accents and beams) but they do not take care of the necessary changes of accidentals when transposing, i.e. the fact that an $F\sharp$ occuring with a zero signature should become a $B\natural$ when transposing from the tonality of C major to F major where the normal B is the $B\flat$. Since the intent of the composer is not obvius – he may want to shift a group of notes within the same tonality or conversely to transpose it in another tonality – this is not done automatically. Thus the `\sh`, `\fl`, `\na`, `\dsh` and `\dfl` symbols *are not affected* by a change of the `\transpose` register.

But the composer/typesetter may ask MusicTEX to do that work. In this case, he should code `\Sh`, `\Fl`, `\Na`, `\dSh` and `\dFl` (or `\bigSh`, `\bigFl`, `\bigNa`, `\bigdSh` and `\bigdFl` or `\smallSh`, `\smallFl`, `\smallNa`, `\smalldSh` and `\smalldFl`) instead of the usual lower case accidental symbols. The symbol `\Sh` (resp. `\bigSh` and `\smallSh`) means that the corresponding pitch has to be raised by *one half pitch* with respect to its normal value *according to the current signature*. Thus `\Sh b` means a $B\sharp$ if the signature is zero or positive, and a $B\natural$ if it is negative. The same logic applies for all accidentals having an upper case forelast letter.

Obviously, the computation is done after taking account of the value of the `\transpose` register.

The compact codes `^`, `_`, `=` are normally not affected by transposition and signatures, but their behaviour can be changed by saying `\relativeaccidentals` and reset by `\absoluteaccidentals` (the default situation).

Although *relative accidental coding* is an easy and safe way of coding *transposable* scores, care should be exercised in getting rid of the habit of saying `\na b` to rise the pitch of a B when the tonality is F major (i.e. with `\signature`n`=-1` or `\generalsignature{-1}`). An example of sophisticated transposition is given in the score `souvenir.tex` (which `\inputs` `souvenix.tex`.

Besides, the typical piano octave transposition *8 - - - - -* can be obtained by coding :

`\octfin` p n

which puts the 8 and dotted line symbols at the pitch p (usually `p` to `r`). The length is n`\noteskip`. This obviously fit for short octaviation denotations. To transpose a whole line, use `\octline` p. Since `\octfin` terminates with a small hook down, to indicate clearly where octaviation stops, you may also like to use `\octsup` which behaves like `\octfin` without the final hook. All this supposes you have an idea of the actual line breaking of your score in that section, and this is admittedly difficult to handle when octave transposition is supposed to long a large number of lines. In that latter case, you can use the `\def\everyline{...}` to insert whatever code you like at each new line of score. This has been used to set octaviation in the score of the *Toccata in F* by Charles-Marie WIDOR[8] : at each bar (or virtual bar, namely `\zbarre`), the \everystaff procedure is updated to produce the convenient code, in case of the line breaking happening before the next definition.

2.11 Ties and slurs

They have been implemented in a way which may look rather ugly, but we think it is the only way of implementing *in one pass* ties and slurs which run *across glue*.

Slurs and ties must be initiated within the pair `\notes...\notes` before the spacing note is coded. They must be terminated also before the last note is coded.

`\itenu` np

(*ten* stands for the italian word *tenuto*) initiates an upper tie (convex) at pitch p. Just like beams, ties have a reference number n, from 0 to 6 (or 9 if `musicadd` is included. `\itenl` np initiates a lower tie (concave).

The tie of reference number n is terminated by `\tten` n.

Slurs are initiated with `\ilegu` np and `\ilegl` np, where *leg* stands for the italian word *legato*, and they are terminated with `\tleg` n. Except that slurs start before the note position and stop after, ties and slurs work the same way with the same syntax. They also share the same registers so that slurs nesting tied notes should have numbers n distinct from each other.

2.12 Bars and spacing

Ordinary *bars* a coded using the macro `\barre` (this is a french word[9]). This macro provides an optional (dis-

[7] This value is saved in another register named `\normaltranspose`.

[8] A French organist (1844-1937) and composer who was in charge of the organ of St-Sulpice in Paris, from 1864 to 1934.

[9] Whose advantage is that it differs from `\bar` which is already defined in TEX.

cretionary) line or page break[10]. It also provides some *glue* in order to expand the text over an evenly filled line.

However, since the number of bars in a score line is generally small, it may be convenient to allow *glue* not only on each sides of *bars*. This can be done using the macro `\temps` (the french word for *beat*). This macro has two effects :

1. it inserts some *glue* but prevents line breaking[11],
2. if some ties or slurs are pending it expands them across the glue by writing an `\hrule` which overlaps the unfinished tie and makes it look prolungated.

Whatever the care you exercize in adjusting the size of the `\elemskip`, you are still likely to find some broken ties (which indicate excessive glue disassembling the gliding tie `\hrule`s) or some unexpected (and unwanted) line breaks or some Over[aw]full `\hbox`es. A useful means of estimating the remaining space to be filled with glue consists in declaring `\raggedlinestrue` : after that, an `\hfil` will be inserted by MusicTEX before each computed (when using `\autolines`) or forced line break. Thus, all the musical text will be packed on the left of the line and you will clearly see the amount of remaining space. Then, it will be up to you do decide changing some spacing parameters.

Important : *do not use* `\temps` *when beams are pending, otherwise their spatial synchronization would fail. In other words, ties and slurs can jump over glue (because horizontal rules may overlap and thus have some elasticity) but beams (as well as any oblique lines) cannot.*

Unless otherwise specified, bars are numbered. This is a good means of finding errors provided that the MusicTEX user has put comments in his source text recording the (expected) bar number. However, this can look unpleasant for final outputs, since the habit is to number bars only each other five or ten bars. This is not a serious problem since the frequency of bar numbering is defined as :

`\freqbarno{1}`

If you replace the `1` by `5`, bar numbering will occur each other five bars. Conversely, whenever you want to putthe bar number, you just say `\wbarno`. You can also inhibit any bar number printing by telling

`\def\wbarno{}`.

The bar counter is also accessible, its name is `\barno`. This has nothing to do the the bar counting inveoked by `\autolines`, so you can change it without any dramatic consequence.

2.13 Line and page breaking

Bars provide a line breaking mechanism which is supposed to enable TEX to break a full score into lines and pages, with an optimal distribution of the text into lines and pages. Unfortunately, this does not work correctly for scores of more that approximately one page. The reason is that TEX must compile the entire contents of a paragraph, before it tries to break it into lines and pages. Therefore, one cannot rely on TEX to make that work which automatically results in the diagnostic `TeX capacity exceeded, memory...`.

To circumvent this dramatically restrictive capacity[12] unless you use some BigTEX[13] another mechanism must be invoked to break lines.

The first one is the manual one : you replace some of the `\barre` macro calls by either `\alaligne` (equivalent to `\break` within text (in fact it contains a `\break` plus some (many) other things. In the same way, you can code `\alapage` to force an `\eject` with proper reinitialization of staffs, clefs and signatures.

The second one is fit for scores whose bars are of regular length : after `\debutmorceau`, you code the following macro :

`\autolines` tml

where t is the number of *elementary spacings* (the lenghth of `\notes...\enotes`) in an average bar, m is the number of bars you wish in a line and l is the number of lines you wish in a page[14].

This sets some parameters, namely `\maxbarsinline` and `\maxlinesinpage` which are simply used to count the bars, optionally perform `\alaligne` or `\alapage` instead of the normal `\barre`. You may freely alter the values of these parameters, once they have been established by `\autolines`. Moreover, you can still force line breaking of page ejection using `\alaligne` or `\alapage` without problem since these macroes actually reset the bar counters appropriately.

On the other hand, you may want to forbide line breaking at a bar, then you replace `\barre` by `\xbarre`.

Conversely, you may want to break a line *not at a bar*[15]. This is allowed by `\zbarre` (optional line break) or forced by `\zalaligne` or `\zalapage`.

The final heavy double bar of a piece is provided by `\finmorceau`. If you just want to terminate

[10] Unless it is triggered according to bar counting when `\autolines` has been invoked.

[11] This could unpleasantly occur if you insert a space...

[12] TEX has been designed to type text, not music.

[13] Whose drawback is that it is very slow on ordinary PCs.

[14] After having coded nearly one hundred of pages of music, I strongly recommend the use of `\autolines` except when inserting short excerpts of less than one line, such as in musicographic books.

[15] For example, you may prefer to turn the page at a place where the pianist has one hand free.

the text with a simple bar, you say `\suspmorceau`. If you want to terminate it without a bar, you code `\zsuspmorceau`. Once you have stopped the score by any of these means, you can restart it using `\reprmorceau`.

If you want the next vertical bar to be a double bar, you have to declare `\setdoublebar` before the `\barre` (or the `\suspmorceau` or `\alaligne` or `\alapage`) to be marked with a double thin bar.

2.14 Changing score attributes

As seen before, you can change the signature of the whole set of instruments by `\generalsignature` n where $n > 0$ means a number of sharps, $n < 0$ means a number of flats. Or, you may prefer to change the signature of only one or two instruments by the statement :

`\signature`r`=`s

where r is the roman numeral of the instrument considered, and s its specific signature. Since you may change simultaneously (with respect to the score) but consecutively (with respect to your code) the signatures of several instruments, this change takes place only when you say `\changesignature` (within a bar) or `\changecontext` (after a single vertical rule) or `\Changecontext` (after a double vertical rule. In the same way, you may want to change the active clefs. This is done by

\cleftoks$r$={{$s1$}{$s2$}{$s3$}{$s4$}}%

where r is the roman numeral of the instrument, $s1$ specifies the clef of the lower staff, $s2$ the clef of the second staff, etc. One must always give four values with the above syntax, otherwise... $s1 = 6$ means the *bass* clef (*clef de fa* in french), $s1 = 0$ means the *violin* clef (*clef de sol* in french), $s1 = 1$ through $s1 = 5$ mean the *alto* clef set on first (lower) through fifth (upper line of the staff). As seen above in the case of signatures, several clefs may be changed at the same time ; thus all the clef changes become operational only when the macro `\changeclefs` is coded. Normal usage consists in issuing this command before the bar, not after (this helps the music player when the change happens across a line break).

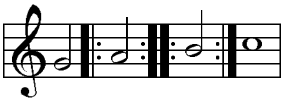
Meter changes are implemented the same way :

`\metertoks`r`={{`$m1$`}{`$m2$`}{`$m3$`}{`$m4$`}}%`

where r is the roman numeral of the instrument, $m1$ specifies the meter of the lower staff, $m2$ the meter of the second staff, etc. One must always give four values with the above syntax, otherwise... Since meter changes are meaningful only across bars, they are actually taken in account with `\changecontext` or `\Changecontext` or `\alaligne` or `\alapage`.

2.15 Repeats

To insert a *repeat bar* you can use several sets of procedures. The simplest consists in using the commands `\leftrepeatsymbol`, `\rightrepeatsymbol` and `\leftrightrepeatsymbol` – coded outside the `\notes...\enotes` pairs – which will simply insert these colon adorned double bars at the requested place. For example :



has been coded as :

```
\notes\hu g\enotes
\leftrepeatsymbol
\notes\hu h\enotes
\leftrightrepeatsymbol
\notes\hu i\enotes
\rightrepeatsymbol
\Notes\wh j\enotes
\suspmorceau
```

However, the previous way of coding does not provide for line breaking at repeat bars, nor does it advance the bar numbering. In fact, this way of coding if only fit for repeats occuring in the middle of a bar.

A second way of coding consists in saying `\setleftrepeat`, `\setrightrepeat` or `\setleftrightrepeat` before a bar (`\barre`), `\suspmorceau` or `\changecontext`). In this case, the next next vertical bar will be replaced with the selected repeat bar. This meets the traditional music typesetting conventions in the only case of the *right repeat* but, unfortunately, left and left/right repeats use to behave in a different manner when in the middle of a line and at a line break.

The third coding, namely the correct coding – i.e. transforming in the correct manner when occuring at a line break – is obtained by substituting the `\barre` command with `\leftrepeat`, `\rightrepeat` and `\leftrightrepeat`.

Now, if you want to force a new line at a repeat, you should code respectively :

```
\setrightrepeat\alaligne
\setrightrepeat\suspmorceau
\alaligne\leftrepeatsymbol
\reprmorceau\leftrepeatsymbol
\debutmorceau\leftrepeatsymbol
```

or the combination of two of these in the case of a left/right repeat.

2.16 Miscellaneous

Special macros are provided to help the composer to set any TEX text on the staffs. The macro

`\charnote` p{ text }

sets the given text with its base line at pitch p of the current staff (this means it must be coded inside `\notes...\enotes`. Whatever the length of the text, the spacing is `\noteskip`. If you do not want it to cause spacing, you code `\zcharnote`. If you want the possible spilling text to expand on the left rather than on the right, then you can use `\lcharnote`.

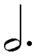
To place some text at the mid-position between the two staffs of a keyboard instrument, you may code :

```
\midtwotext{ text }  % (spacing)
\zmidtwotext{ text }  % (non spacing)
```

being however careful, a) to put it inside `\notes...\enotes`, b) to code it in the text of the lower staff.

A text to be put above the current staff is introduced by `\uptext{...}`. This may however cause some collision with bar numbering or notes above the staff ; it is then wise to use `\Uptext{...}` which puts the text two pitches higher (recommended to post the tempo).

Metronomic indication deserves a special macro. The mention :

 = 60

is coded by `\metron{\hup}{60}` (normally embedded in `\Uptext` which is in turn embedded within `\notes...\enotes`).

The macro `\zcharnote` is fit for coding special notations like accents above or below the notes. It behaves like `\charnote` but causes no spacing.

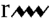
Arpeggios (i.e.) can be coded with the macro

`\arpeggio` pm

where p is the pitch of the base of the arpeggio symbol and m is its multiplicity (one period is equal to one space between staff lines, i.e. 5 points). This macro causes a space of one note head width. If should be issued before the concerned chords. Its variant `\larpeggio` sets the arpeggio symbol slightly more on the left, in order to avoid collision with accidentals in front of the chords.

Trills can be coded in several ways. `\trille` l (where l is a TEX dimension) yields while `\Trille` l (where l is a TEX dimension) yields *tr* . To put these patterns at a given pitch, one may use `\xtrille` pl or `\xTrille` pl. On the other hand `\ntrille` pn is equivalent to `\xtrille` p{n`\noteskip`} and `\nTrille` pn is equivalent to `\xTrille` p{n`\noteskip`}

Other *ornaments* are available :

- `\mordant` p for ,
- `\pince` p for ,
- `\Pince` p for  ,
- `\upz` p (upper *pizzicato*) to put a dot above a note head at pitch p,
- `\lpz` p (lower *pizzicato*) to put a dot below a note head at pitch p,
- `\usf` p (upper *sforzando*) to put a $>$ accent above a note head at pitch p,
- `\lsf` p (lower *pizzicato*) to put a $>$ accent below a note head at pitch p,
- `\ust` p (upper *staccato* or *portato*) to put a hyphen above a note head at pitch p,
- `\lst` p (lower *staccato* or *portato*) to put a hyphen below a note head at pitch p,
- `\Upz` p (upper strong *pizzicato*) to put an apostrophe above a note head at pitch p,
- `\Lpz` p (lower strong *pizzicato*) to put a reversed apostrophe below a note head at pitch p.

The procedure named `\everystaff` is executed each time a new system is typed. It is normally void, but it can be defined (simply by `\def\everystaff{...}`) to tell MusicTEX to post anything reasonable at the beginning of each system.

The procedure named `\atnextline`, normally void, is executed at the next computed (through `\autolines`) or forced line break (using `\alaligne` or `\alapage`). More precisely, it is executed after the break and before the next system is typed. Thus it is fit for posting new definitions of layout parameters, when no system is pending[16].

2.17 Staff size

You may also want to write some parts of your score in 20pt staff size and in 16pt staff size. This cannot be done – at least presently – in the same system, but only for distinct parts of pieces. Changing the size is done by saying :

`\musicsize=16\computespecifics` or

`\musicsize=20\computespecifics`

respectively.

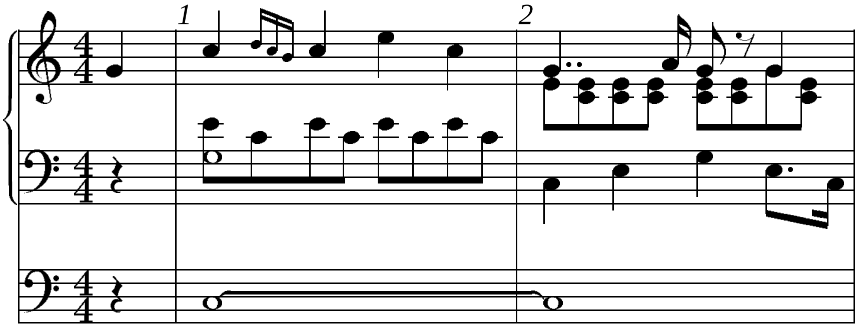
2.18 Small and tiny notes

Ornaments and *cadenzas* usually need to be written using smaller notes[17]. This can be done everywhere by stating `\smallnotesize` or `\tinynotesize`. Normal note size is restored by `\normalnotesize`.

These macros only have a local scope. Thus, if these macros are invoked outside the `\notes...\enotes` pair, the change is valid for the rest of the piece unless explicitely modified but, if they are invoked inside, their effect is local to the current staff of the current `\notes...\enotes` pair. As an example, the following excerpt (beginning of the Aria of the Creation by Joseph HAYDN)

[16] Its logic is similar to plain TEX's `\vadjust` command.

[17] This is independent of the staff size.



can be coded as :

```
\def\DS{\hbox{\ds}}
\def\FS{\hbox{\kern 0.3\noteskip\soupir}\kern -0.3\noteskip}
\def\qbl#1#2#3{\ibl{#1}{#2}{#3}\qb{#1}{#2}}%
\def\qbu#1#2#3{\ibu{#1}{#2}{#3}\qh{#1}{#2}}%
\def\nbinstruments{2}%
\generalmeter{\meterfrac{4}{4}}%
\signaturegenerale{0}%
\nbporteesii=1\relax
\nbporteesii=2\relax
\cleftoksi={{6}{0}{0}{0}}
\cleftoksii={{6}{0}{0}{0}}
\etroit
%
%  end of preliminary definitions
%
\debutextrait
\NOTes\soupir&\soupir|\qu g\enotes
% mesure 1
\advance\barno by -1\relax
\barre\NOtes\itenu2J\wh J&\zw N\ibl0c0\qb0e|\qu j\enotes
\notes&\qbl0c0|\noteskip=0.6\elemskip\tinynotesize
\Ibbu1ki2\qh1{kj}\tqh1i\qsk\enotes
\Notes&\qb0e\tbl0\qb0c|\qu j\enotes
\temps\Notes&\ibl0c0\qb0{ece}\tbl0\qb0c|\ql l\sk\ql j\enotes
% mesure 2
\barre\Notes\tten2\wh J&\ql J\sk\ql L|\ppt g\rlap{\qu g}\qbl1e0\relax
       \zq c\qb1e\zq c\qb1e\relax
       \zq c\tbl1\rlap{\qb1e}\ \ \ccu h\enotes
\temps\Notes&\ql N\sk\pt L\ibl0L{-4}\qb0L|\ibl1e0\zq c\rlap{\qb1e}\cu g\relax
       \zq c\rlap{\qb1e}\raise\Interligne\DS \rlap{\qu g}\qb1g\enotes
\notes&\sk\tbbl0\tbl0\qb0J|\tbl1\zq c\qb1e\enotes
\finextrait
```

2.19 Layout parameters

Most layout parameters are set by MusicTEX to reasonable default values. However, sophisticated scores[18] may need more place below the lowest staff, between staves, etc.

`\interligne`: vertical interval between lines

`\nullthick`: reserved height above base line for zero staff lines (text of songs)

`\bottommargin`: margin below the first staff of the lowest instrument. If not assigned a non zero dimension, it is set to `\bottomfacteur\Interligne` at the next system.

`\topmargin`: margin above the upper staff ot the upper instrument

`\internote`: the half of `\Interligne`

`\interbeam`: vertical distance between beams

`\Interligne`: vertical distance between the base of staff lines (`\interligne` is the size of the blank space between lines. The difference is the line thickness `\lthick`.

[18] To our knowledge, the most complicated scores are those written for the piano, during the romantic and post-romantic periods.

`\interportee:` the distance between the top of one staff and the bottom of the next one. If not assigned a non zero value, it is set to `\interfacteur\Interligne` at the next system.

`\interinstrument:` the additional vertical distance between two different instruments. This means that the distance between the upper staff of the previous instrument and the lowest line of the current instrument is equal to `\interportee+\interinstrument`. This value is normally zero, but it helps putting additional space between distinct instruments for the sake of clarity. This is a general dimension which holds for each of the vetical spaces between instruments, except the upper one, in which case this interval is irrelevant. However, this parameter can be overriden for the space above a specific instrument. For example (see the example `angescao.tex`) one can state :

```
\def\interinstrumenti{5pt}
```

to force an additional spacing of 5 points above instrument i, whatever the value of `\interinstrument`.

`\systemheight:` the distance from the bottom of the lowest staff to the top of the highest staff of the upper instrument. This is the height of the vertical bars (single, double, repeats, etc.).

In addition, when handling notes of a given staff of a given instrument, the following dimensions are available (note these are not true registers, but *equivalenced symbols* through a `\def`) :

- `\altitude` : the altitude of the lowest line of the lowest staff of the current instrument.
- `\altportee` : the altitude of the lowest line of the current staff.

Most of these values can be changed, but only between the end of the previous system and the beginning of the next one. This can be inserted between a `\suspmorceau` and a `\reprmorceau`, but it is wiser to say, for example :

```
\def\atnextline{\global %
\bottommargin=5\Interligne}
```

When doing so, the MusicTEX user should be aware that this could disturb pending *slurs* or *ties*, since the altitude of these is stored in an absolute way, starting from the baseline of the systems. Therefore, changing the `\bottommargin` dimension can also be made by means of `\advancebottom{`$< dimension >$`}` which updates all pending slur and tie altitudes by the given dimension. This has been used in `pacifiqn` and `pacifiqb`.

2.20 MusiclaTEX

As said before, the amount of memory used by MusicTEX makes it hardly compatible with LaTEX. However, Nicolas BROUARD succeeded in building a `\musictex.sty` which is now included in the distribution. This is not recommended to make separate music scores. Its purpose is rather to provide a means of inserting short musical excerpts in books or articles witten with latex. Then, the `\documentstyle` command should include `musictex` in the options.

The LaTEX style file `musictex.sty` is the simple concatenation of the following files :

- `musicpre.tex`
- `musicnft.tex`
- `musictex.tex`
- `musicvbm.tex`
- `musicpos.tex`

In the case of a LaTEX user wanting to use accidental transposition facilities, he should invoke `musictrp` in the options of the `\documentstyle` command. The `musicpos.tex` file merely overrides the `\catcodes` of the `|` and `&` symbols which are modified by MusicTEX. To have access to these symbols when coding music, on should then enclose the scores or excerpts within `\begin{music}` and `\end{music}`. But there is also another possibility, i.e. to say `\nextinstrument` instead of `&` and `\nextstaff` instead of `|`. And in case of `TeX capacity exceeded...`, use a “BiglaTEX” (after checking there is no visible error in the source code).

2.21 Implementation and restrictions

The macroinstruction file MusicTEX contains approximately 2500 lines of code, that is 80 000 bytes approximately. This requires your score to be compiled by the most extended versions of TEX (65 000 words of working memory). It is therefore wise to set `\tracingsstats` to 2 in order to have an information about the memory used in each page. In desperate situations, we recommend using the “BigTEX” processors which, unfortunately, perform a great deal of disk input/outputs (on PCs) which make them awfully slow.

In particular, the number of registers it uses and the amount of memory used by LaTEX macros makes it doubtfully compatible with LaTEX, unless using BigLaTEX.

Other precautions are necessary : beware of end-of-line spaces ; they corrupt layout and may cause unwanted line breakings after which music symbols seem to *float* in the air without staffs. To avoid that, it is recommended to use `\relax` rather than `%` at the end of source lines, since it allows indentation at the following source line.

3 Installation

As seen before, all the files are available at *anonymous ftp* `rsovax.ups.circe.fr` (130.84.128.100) in the directory

`[anonymous.musictex]`.

This directory normally contains `musictex.zip` which contains all the distribution for PC (MS-DOS) computers. This is only for `ftp`-ing convenience since all source files are directly available in the same directory.

The VMS files are also packed into `musictex.bck`. Notwithstanding the fact that files are packed together or not, the files provided are of two kinds :

1. the basic files ;
2. the example files.

All *basic files* are either of the form `music*.*` (exclusing of course `*.zip` and `*.bck`) and `beamn*.*`. Other files (`*.tex` or `*.dvi`) are example files.

Fonts are provided as `*.mf` files but also as `*.tfm` and `*.pk` files for 300 dpi printers or previewers. Additional values of the dpi parameter are provided in `musicpk.zip`.

REMARK : introducing these files in a *format* (with INITEX) is a means of saving computer time.

4 Examples

Three examples are displayed in the following pages, namely a transcription for organ and song of the beginning of the *Aria* of *The Creation* by Joseph Haydn, a *motet* for Christmas by Praetorius – in the original pitch and transposed down by one pitch in smaller size – and a transcription of a concerto for oboe by Benedetto Marcello.

REMARK : the examples are given at the end of the document because their require a BigLaTEX to be produced. Thus a trial TEX-ing with standard LaTEX can produce anything but the large examples (they can be more easily produced using *plain TEX* rather than LaTEX.

People who want to produce some of the given examples sould take care of the fact that many given files are supposed to be included (by means of `\input`) in other files. Thus the good files to be directly TEX-ed are those which begin with \input musicnft.

Suggested tests are :

- `PACIFIQN` for a long original piano work (11 pages) ;
- `HADAGIM` for a typical organ score ;
- `CARILLON` for a sophisticated piano score (use BigTEX !) ;
- `PRAETORI` to get an ancient polyphonic song with three transpositions ;
- `ANGESCAM` if you like Christmas carols with four voices, a three-staff organ score and the same transposed to meet ordinary singer's limitations ;
- `HPRELFUG` if you like imitations of J.-S. Bach...
- `MARCELLO` if you likepre-baroque musicé;
- `RECUEIL` if you want to get all the organ works of the author in a single book (43 pages) ;
- `HCREATIM` to get an unfinished transcription of the Creation by J. HAYDN.
- `HWIDOR` and `NWIDOR` to get the Toccata by Charles-Marie WIDOR in two different sizes.

```
\input musicnft
\input musictex
\input musicsty

\def\nbinstruments{2}%
\generalmeter{\meterfrac{4}{4}}%
\signaturegenerale{0}%
\hsize   190mm     \vsize 275mm
\hoffset -10mm     \voffset -10mm
\centerline{\enorme Aria No. 24}
\medskip\centerline{\moyen (La Cr\'eation)}
\medskip\centerline{\moyen Joseph HAYDN}
%
\rightline{Transcription pour Orgue et T\'enor, D. Taupin (1990)}
\medskip
\nbporteesi=1\relax
\nbporteesii=2\relax
\def\qbl#1#2#3{\ibl{#1}{#2}{#3}\qb{#1}{#2}}%
\def\qbu#1#2#3{\ibu{#1}{#2}{#3}\qh{#1}{#2}}%
\etroit
\def\DS{\hbox{\ds}}\def\FS{\hbox{\kern 0.3\noteskip\soupir}\kern -0.3\noteskip}
\cleftoksi={{6}{0}{0}{0}}
\cleftoksii={{6}{0}{0}{0}}
\debutmorceau
\NOTes\soupir&\rlap{\rmidtwotext{\bf II}}\soupir|\qu g\enotes
% mesure 1
\advance\barno by -1\relax
\barre\NOtes\itenu2J\wh J&\zw N\ibl0c0\qb0e|\itenl0j\ibu1l0\qh1j\enotes
\notes&\qbl0c0|\nbbu1\nbbbu1\tten0\qh1{jkj}\tbu1\qh1i\enotes
\Notes&\qb0e\tbl0\qb0c|\qu j\enotes
\temps\Notes&\ibl0c0\qb0{ece}\tbl0\qb0c|\ql l\sk\ql j\enotes
% mesure 2
\barre\Notes\tten2\wh J&\ql J\sk\ql L|\ppt g\rlap{\qu g}\qbl1e0\relax
       \zq c\qb1e\zq c\qb1e\relax
       \zq c\tbl1\rlap{\qb1e}\ \ \ccu h\enotes
\temps\Notes&\ql N\sk\pt L\ibl0L{-4}\qb0L|\ibl1e0\zq c\rlap{\qb1e}\cu g\relax
       \zq c\rlap{\qb1e}\raise\Interligne\DS \rlap{\qu g}\qb1g\enotes
\notes&\sk\tbbl0\tbl0\qb0J|\tbl1\zq c\qb1e\enotes
% mesure 3
\barre\NOtes\itenl2G\wh G&\zw N\raise 3.5\Interligne\ds
|\rlap{\cl f}\itenl0k\ibu1m0\qh1k\enotes
\notes&\qbl0b0|\nbbu1\nbbbu1\tten0\qh1{klk}\tbu1\qh1{^j}\enotes
\Notes&\zq d\qb0f\tbl0\qb0b|\qu k\enotes
\temps\Notes&\ibl0d0\zq d\qb0{fb}\zq d\qb0f|\qu m\sk\pt k\qbu1k{-4}\enotes
\notes&\tbl0\qb0b|\sk\tbbu1\tbu1\qh1i\enotes
% mesure 4
\alaligne\Notes\tten2\wh G&\ql G\sk\ql I|\rlap{\qupp g}\ibl1c0\qb1g\relax
       \zq{bd}\qb1f\zq{bd}\qb1f\relax
       \zq{bd}\tbl1\rlap{\qb1f}\ \ \ccu h\enotes
\temps\Notes&\ql K\sk\pt I\qbl0I{-4}|\ibl1d0\rlap{\qb1b}\cu g\relax
       \zq{bd}\rlap{\qb1f}\raise\Interligne\DS \rlap{\qu g}\qb1g\enotes
\notes&\sk\tbbl0\tbl0\qb0G|\tbl1\zq{bd}\qb1f\enotes
% mesure 5
\barre\Notes\hu J&\rlap{\lhu J}\ibl0M0\qb0J\zq N\qb0c\zq N\qb0c\tbl0\zq N\qb0c\relax
      |\rlap{\hl e}\qu j\sk\qbu1l{-4}\tbu1\qh1j\enotes
\temps\Notes\hu K&\rlap{\lhu K}\ibl0M0\qb0K\zq N\qb0b\zq N\qb0b\tbl0\zq N\qb0b\relax
      |\rlap{\hl f}\ibu1k0\qh1{ikm}\tbu1\qh1k\enotes
% mesure 6
\barre\Notes\wh L&\zw N\raise 3.5\Interligne\ds\qbl0c0\qb0e\relax
      |\rlap{\hl g}\ppt j\qu j\enotes
\notes&\tbl0\qb0c|\sk\ccu l\enotes
\temps\Notes&\ibl0c0\qb0{ece}|\rlap{\hl g}\qu n\sk\raise 2\Interligne\DS\enotes
\notes&\tbl0\qb0c|\ibbu1m{-3}\qh1m\tbu1\qh1l\enotes
```

music example

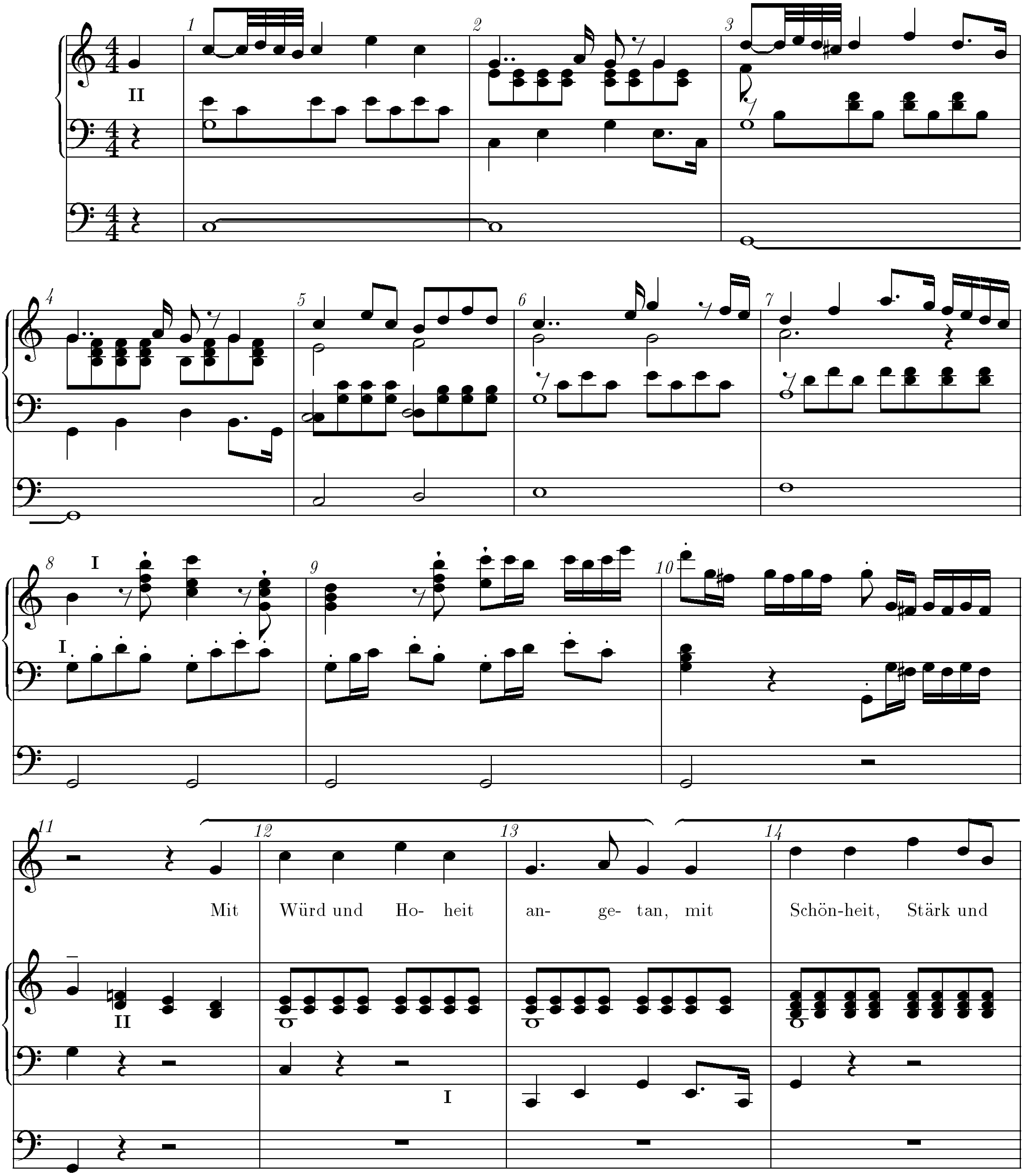
```
% mesure 7
\barre\Notes\wh M&\zw a\raise 3.5\Interligne\ds\qbl0d0\qb0f\tbl0\qb0d\relax
      |\rlap{\hlp h}\qu k\sk\qu m\enotes
\temps\Notes&\ibl0d0\qb0f|\pt o\qbu1o{-3}\enotes
\notes&\zq d\qb0f\sk
\zq d\qb0f\sk\tbl0\zq d\qb0f|\sk\tbbu1\tbu1\qh1n\relax
      \ibbu1m{-3}\rlap{\raise -\Interligne\qp}\qh1{mlk}\tbu1\qh1j\enotes
% mesure 8
\alaligne\Notes\hu G&\lcharnote c{\bf I}\relax
\pz b\ibl0M3\qb0N\pz d\qb0b\pz f\qb0d\pz d\tqb0b|\ql i\rlap{\uptext{\bf I}}\sk\ds\ppz p\zq{km}\cl
\temps\Notes\hu G&\ibl0M3\pz b\qb0N\pz e\qb0c\pz g\qb0e\pz e\tqb0c\relax
        |\zq{jl}\ql q\sk\ds\ppz l\zq{gj}\cl l\enotes
% mesure 9
\barre\Notes\hu G&\pz b\ibl0M3\qb0N|\zq{gi}\ql k\enotes
\notes&\nbbl0\qb0b\tqb0c\enotes
\zglu\Notes&\pz f\ibl0d{-4}\qb0d\pz d\tqb0b|\ds\ppz p\zq{km}\cl p\enotes
\temps\Notes\hu G&\pz b\ibl0M3\qb0N|\ppz q\ibl1m3\zq l\qb1q\enotes
\notes&\nbbl0\qb0c\tqb0d|\nbbl1\qb1q\tqb1p\enotes
\temps\notes&\pz g\ibl0e{-4}\qb0e\sk\pz e\tqb0c|\ibbl1p0\qb1{qpq}\tqb1s\enotes
% mesure 10
\barre\Notes\hu G&\zq{Nb}\ql d|\pz t\ibl1o{-3}\qb1r\enotes
\notes&|\nbbl1\qb1n\tqb1{ˆm}\enotes
\zglu\notes&\soupir|\ibbl1m0\qb1{nmn}\tqb1m\enotes
\temps\Notes\hpause&\ibl0I6\pz J\qb0G|\pz p\cl n\enotes
\notes&\nbbl0\qb0N\tqb0{ˆM}|\ibbu1g{-3}\qh1g\tqh1{ˆf}\enotes
\notes&\ibbl0N0\qb0{NMN}\tqb0M|\ibbu1g0\qh1{gfg}\tqh1f\enotes
\suspmorceau
\def\nbinstruments{4}%
\nbporteesiii=0\relax
% mesure 11
\reprmorceau
\NOtes\qu G&\ql N|\st n\qu g&&\hpause\enotes
\zglu\NOtes\soupir&\rlap{\rmidtwotext{\bf II}}\soupir|\zq d\qu{=f}&&\enotes
\temps\NOtes\hpause&\hpause|\zq c\qu e&&\soupir\enotes
\zglu\NOtes&|\zq b\qu d&Mit˜&\ilegu0p\qu g\enotes
% mesure 12
\barre\Notes&\qu J|\zw N\zq c\ibu0e0\qh0e\zq c\qh0e&W\"urd˜&\ql j\enotes
\Notes&\soupir|\zq c\qh0e\zq c\tqh0e&und˜&\ql j\enotes
\temps\Notes\pause&\hpause|\zq c\ibu0e0\qh0e\zq c\qh0e&Ho-&\ql l\enotes
\Notes&\zcharnote C{\bf I}|\zq c\qh0e\zq c\tqh0e&heit˜&\ql j\enotes
% mesure 13
\barre\Notes&\qu C|\zw N\zq c\ibu0e0\qh0e\zq c\qh0e&an-&\qup g\enotes
\Notes&\qu E|\zq c\qh0e\zq c\tqh0e&\sk ge-&\sk\cu h\enotes
\temps\Notes\pause&\qu G|\zq c\ibu0e0\qh0e\zq c\qh0e&tan,˜&\tleg0\qu g\enotes
\Notes&\ibu1E{-3}\qhp1E|\zq c\qh0e&mit˜&\ilegu0p\qu g\enotes
\notes&\sk\tbbu1\tqh1C|\zq c\tqh0e&&\enotes
% mesure 14
\barre\Notes&\qu G|\zw N\ibu0e0\zq{bd}\qh0f\zq{bd}\qh0f\relax
    &Sch\"on-&\ql k\enotes
\Notes&\soupir|\zq{bd}\qh0f\zq{bd}\tqh0f&heit,˜&\ql k\enotes
\temps\Notes\pause&\hpause|\zq{bd}\qh0f\zq{bd}\qh0f&St\"ark˜&\ql m\enotes
\Notes&|\zq{bd}\qh0f\zq{bd}\tqh0f&und˜&\ibu2k{-4}\qh2k\tqh2i\enotes
\suspmorceau
\removelastskip\rightline{\sl\aujourdhui}
\bye
```

Aria No. 24

(La Création)

Joseph HAYDN

Transcription pour Orgue et Ténor, D. Taupin (1990)


1
2
3
II
4
5
6
7
8
I
I
9
10
11
12
13
14
Mit Würd und Ho- heit an- ge- tan, mit Schön-heit, Stärk und
II
I

20 avril 1994

Dating with TeX

Theo A. Jurriens

Kapteyn Astronomical Institute
P.O. Box 800
9700 AV Groningen
The Netherlands

taj@rugr86.rug.nl

Abstract

Three TeX-coded algorithms are given for performing tricks with dates.

1 Introduction

In this article we give a method for calculating the so-called Julian Date. This date is used in astronomy to avoid problems in counting with leap years, days etc. The Julian Date starts every day at noon.

Three basic algorithms are TeX-coded:

- from calendar date (Gregorian) to Julian Date. Only dates after 1582 October 15 are valid,
- from Julian Date to calender date,
- and the day of week

The algorithms and some examples used can be found in Chapter 3 of the book ‘Astronomical Formulae for Calculators’ by Jean Meeus (Willmann-Bell Inc. 1988). The TeX-coded algorithms are useful in different daily-life applications.

2 Macro’s

Macros are given below, using the as
`\input kalender.sty` or
`\documentstyle[kalender]`.

```
%%%%%%%%%%%%%%%%%%%%%%
%%% kalender.sty   %%%
%%%%%%%%%%%%%%%%%%%%%%
  \newcount\a
  \newcount\alfa
  \newcount\b
  \newcount\c
  \newcount\d
  \newcount\dag
  \newcount\dg
  \newcount\dw
  \newcount\e
  \newcount\f
  \newcount\jaar
  \newcount\jd
  \newcount\jr
  \newcount\klad
  \newcount\m
  \newcount\maand
  \newcount\mnd
  \newcount\x
  \newcount\y
  \newcount\z
%%%
\def\datjul#1#2#3{
\jaar=#1
\maand=#2
\dag=#3
\ifnum\maand>2
   \y=\jaar
   \m=\maand
\else
   \y=\jaar \advance\y by-1
   \m=\maand \advance\m by12
\fi
\a=\y
\divide\a by100
\b=2
\advance\b by-\a
\x=\a
\divide\x by4
\advance\b by\x
%
\jd=\y
\multiply\jd by36525
\divide\jd by100
%
\x=\m
\advance\x by1
\multiply\x by306001
\divide\x by10000
\advance\jd by\x
\advance\jd by\dag
\advance\jd by1720995
\advance\jd by\b
}
%
\def\dayofweek#1#2#3{\datjul{#1}{#2}{#3}
\x=\jd
\advance\x by 1
\dw=\x
\divide\x by 7
\multiply\x by 7
\advance\dw by -\x
}
%
%
\def\juldat#1{
\z=#1
\ifnum\z<2299161
   \a=\z
\else
   \alfa=\z
   \multiply\alfa by100
   \advance\alfa by-186721625
   \divide\alfa by3652425
   \a=\z
   \advance\a by1
   \advance\a by\alfa
   \divide\alfa by4
   \advance\a by-\alfa
\fi
%
\b=\a
\advance\b by1524
%
\c=\b
\multiply\c by100
\advance\c by-12210
\divide\c by36525
%
\d=\c
\multiply\d by36525
\divide\d by100
%
\e=\b
\advance\e by-\d
\multiply\e by10000
\divide\e by306001
%
%
\x=\b
\advance\x by-\d
\dg=\x
\x=\e
\multiply\x by306001
\divide\x by10000
\advance\dg by-\x
%
\mnd=\e
\advance\mnd by-1
\ifnum\e>13
    \advance\mnd by-12
\fi
%
\jr=\c
\advance\jr by-4715
\ifnum\mnd>2
   \advance\jr by-1
\fi
}
%
```

The table below illustrates the use of the macro's.

macro	output variable
`\datjul{year}{month}{day}`	`\jd`
`\juldat{julian_date}`	`\jr \mnd \dg`
`\dayofweek{year}{month}{day}`	`\dw`
	0 = sunday, 1 = monday, 2 = tuesday, 3 = wednesday,
	4 = thursday, 5 = friday , 6 = saturday

3 Examples

The date of the launch of the first Sputnik corresponds to a Julian date of 2436116.

```
The date of the launch of the first Sputnik
corresponds to a Julian date of
\datjul{1957}{10}{4} \number\jd .
```

The second Russian Revolution took place at 1991, August 19:

`\dayofweek{1991}{8}{19} \number\dw` ⟶ 1
a Monday.

The macro's can also be used to calculate date differences.

```
\datjul{1961}{9}{9}
\global\advance\jd by-150
\juldat{\number\jd}
\number\jr\ \number\mnd\ \number\dg .
```

1961 4 12. This date is corresponding with the date of the launch of the first man into space: Yoeri Gagarin. In other words 150 days before the birth of the author.

```
\newcount\klad
\datjul{1992}{3}{15}
\klad=\jd
\datjul{1961}{9}{9}
\advance\klad by-\jd
\number\klad
```

At the moment of writing the author is 11145 days old.

In the appendix you find an other dating example, calculating the dates of easter, the algorithms can be find also in the book of Meeus.

A Date of Easter

```
\newcount\x
\newcount\klad
\newcount\a
\newcount\b
\newcount\c
\newcount\d
\newcount\e
\newcount\f
\newcount\g
\newcount\h
\newcount\i
\newcount\j
\newcount\k
\newcount\l
\newcount\m
\newcount\n
\newcount\o
\newcount\p
\newcount\start
\newcount\einde
\def\pasen#1{
\x=#1
\a=\x
\divide\a by 19
\klad=\a
\multiply\klad by 19
\a=\x
\advance\a by -\klad
\b=\x
\divide\b by 100
\klad=\b
\c=\x
\multiply\klad by 100
\advance\c by -\klad
\d=\b
\divide\d by 4
\e=\b
\klad=\d
\multiply\klad by 4
\advance\e by -\klad
\f=\b
\advance\f by 8
\divide\f by 25
\g=\b
\advance\g by -\f
\advance\g by 1
\divide\g by 3
\h=\a
\multiply\h by 19
\advance\h by \b
\advance\h by -\d
\advance\h by -\g
\advance\h by 15
\klad=\h
\divide\h by 30
\multiply\h by 30
\advance\klad by -\h
\h=\klad
\i=\c
\divide\i by 4
\k=\c
\klad=\i
\multiply\klad by 4
\advance\k by -\klad
\l=32
\klad=\e
\multiply\klad by 2
\advance\l by \klad
\klad=\i
\multiply\klad by 2
\advance\l by \klad
\advance\l by -\h
\advance\l by -\k
\klad=\l
\divide\l by 7
\multiply\l by 7
\advance\klad by -\l
\l=\klad
\m=\a
\klad=\h
\multiply\klad by 11
\advance\m by \klad
\klad=\l
\multiply\klad by 22
\advance\m by \klad
\divide\m by 451
\n=\h
\advance\n by \l
\advance\n by 114
\klad=\m
\multiply\klad by -7
\advance\n by \klad
\klad=\n
\divide\n by 31
\p=\n
\multiply\p by 31
\advance\klad by -\p
\p=\klad
\advance\p by 1
%
\hbox to 3cm{
\the\x %
\ifnum\n=3    %
      \ \ March %
        \else %
      \ \ April %
\fi %
\hfill \the\p}%
\hfill\break
}
%starting year after 1583}
\start=1992
%end year
\einde=2020
%
%
\loop\ifnum\start<\einde
\pasen{\start}
\advance\start by 1
\message{\the\start}
\repeat
```

```
1992 April   19

1993 April   11
1994 April    3
1995 April   16
1996 April    7
1997 March   30
1998 April   12
1999 April    4
2000 April   23
2001 April   15
2002 March   31
2003 April   20
2004 April   11
2005 March   27
2006 April   16
2007 April    8
2008 March   23
2009 April   12
2010 April    4
2011 April   24
2012 April    8
2013 March   31
2014 April   20
2015 April    5
2016 March   27
2017 April   16
2018 April    1
2019 April   21
```

Dag van het Document

Verslag informatiedag ITI-TNO

Kees van der Laan

24 Februari 1992

1 Achtergrond

Het waren er twee: de eigenlijke dag met lezingen en een beurs, en de volgende ochtend met een workshop. De organisatie lag in handen van ITI-TNO, met Joop van Gent, als coördinator.

2 Lezingen

De leidraad van de lezingen was de opslag en retrieval van documenten, en algemener hoe gaan wij om met de zwellende informatiestromen. Joop van Gent schetste de context. Een stijds nijper wordend probleem tekent zich af: het wordt ons allemaal teveel, wij worden overspoeld door de berg van papier en de minder zichtbare maar oh-zo dwangmatige electronische informatie. Kunnen wij het met behulp van computers hanteerbaar houden? Of moeten wij maar rücksichtslos opslaan en terugzoeken wanneer nodig, dit na selectie? De automatische bibliothecaris? Of ontstaat er het beroep van IT-keurder?

Godfrey Smart sprak zeer meeslepend onder de titel Document Management and Retrieval.

Hilde van der Tocht deed weer eens uit de doeken hoe het productieproces bij Elsevier Science Publishers verloopt, onder de titel: Van manuscript tot electronisch product.

David Evans ging er puur taal-technisch tegenaan. Dit was meer een stand van zaken m.b.t. de research. Het draaide om het 'partoon' herkennen, het automatisch herkennen van inhoudelijke aspecten uitgaande van de syntaxis en de semantiek. Ambitieus.

Gerard Kempen sprak over intelligente auteursomgevingen. Zijn groep houdt zich bezig met de ontwikkeling van hulpmiddelen zoals: Norma—ter ondersteuning van huisstijlen, normalisatie en consisientie, Corrie—spellingschecker met kennis van woord- en zinstructuur.

In Libraries without walls, werd gerapporteerd over het CMU/DEC project om bibliotheken via de (huis)computer te raadplegen. Interessant ook vanwege de consequenties t.a.v. toegang: geen tijds- noch afwezigheidsrestricties. Dit laatste geldt zowel t.a.v. personeel als het materiaal zelf.

Ikzelf gospelde een ander maal over SGML en TEX, m.n. over de relatie tussen beiden. Benadrukt werd dat de transformatie van wiskunde en tabellen in zijn algemeenheid een probleemgebied is, en dat voorlopig auteurs naast schrijver ook programmeur moeten zijn voor een perfecte opmaak.[1]

Bronkhorst sprak over de verwerkingsachterstand bij ministeries, en vooral over het hoe om te gaan met de menselijke aspecten bij de invoering van kantoorautomatisering.

De dag werd afgesloten met een forumdiscussie, helaas was deze wat flauw. De bitterballen en de drankjes waren daarentegen pittig en des te lekkerder.

2.1 Indruk

Mijn persoonlijke mening is dat er niet gestreefd moet worden naar het 'alles' bewaren, niet ons door de angst—het zou weleens belangrijk kunnen worden—laten meeslepen.

Wij moeten de computer—de gebruikers ervan dus—leren, het niet-relevante (automatisch) te vergeten. Dit roept een schijnbaar selectieprobleem op. De context van betrokkenen en hun onderlinge verhoudingen is een graadmeter voor relevantie. Ik vond het vreemd dat het wetenschappelijke model van samenvattingen en trefwoorden, c.q. betreffende/kenmerk bij brieven, en het hele referee mechanisme, onderbelicht bleven. Misschien omdat voor algemene documenten er geen classificatiesysteem bestaat?

3 De beurs

De beurs was zeer goed bezet met een 20-tal stands. Wij (NTG) hadden ons bescheiden PR-materiaal in eenvoudige mapjes op een tafel gelegd.

Stands/bedrijfjes:

- **CELEX**
  Centre for Lexical information. Lexicografische

[1] Afgezien van de rol van uitgevers.

database voor Nederlands, Engels en Duits. Zij hebben een nieuwsbrief (gratis).

- **Cognitech**
  Language checking tools for Dutch.
- **Customized Publishing**
  VSR en Océ. Zij hanteren Wordperfect en SGML.
- **Druide**
  Document Retrieval Using Information from Document Elements.
- **Profglot**
  A multi-lingual natural processor, geprogrammeerd in prolog.
- **SPIN**
  Human-Computer Communication using natural language. Een groot project met druide als onderdeel.
- **Spirit**
  Een hulpmiddel voor ‘schrijvers.’ Het bevat een tekstverwerker, planningshulpmiddel, en een opslaggereedschap (database?) voor document onderdelen.

4 Conclusie

Ik vond Corrie en Norma aardig en realistisch.[2] Een ding is vooral duidelijk geworden: er is veel gaande rondom computer-assisted writing, vooral vanuit de taalkundige hoek. Een weelde aan nieuwe informatie werd over het voetlicht gebracht. De echte IT is aan het opstaan. Iets voor de NTG, een beurs tijdens de bijeenkomsten?

ITI-TNO verdient een pluim met de organisatie van dit gebeuren en het vlekkeloos laten verlopen ervan. Een soort proceedings, met extended abstracts staat er aan te komen.

DocumentTEXnisch gesproken is het al heel gewoon dat vrij *complexe* kopij electronisch overgestuurd en verwerkt wordt. Mijn verhaal was een bewijs daarvoor. Hoezo uitwisseling van documenten? Just do it!

[2] Theo beinvloedt mij toch meer dan ik dacht.

Molecuul Muis Manuscript

Verslag KNCV symposium

Gerard van Nes

18 oktober 1991

Abstract

Op 18 oktober 1991 werd door de sectie Computertoepassingen van de KNCV (Koninklijke Nederlandse Chemische Vereniging) een symposium m.b.t. chemische tekstverwerking georganiseerd. Naast een algemene introductie over 'Electronisch Publiceren', kregen zowel de Apple Macintosh als MS-DOS geïnteresseerden (chemici) de huidige mogelijkheden van het verwerken van chemische teksten voorgeschoteld. Het pakket TEX kwam in een laatste lezing naar voren. De duidelijk geslaagde dag werd bezocht door een kleine honderd deelnemers, inclusief een tiental leveranciers. Vele (leerzame) demonstraties, zowel tijdens de lezingen, als ook tussen de lezingensessies door, zorgden mede voor een duidelijk overzicht.

1 Algemeen

Chemische tekstverwerking is eigenlijk een wereld apart. Vooral de grote verscheidenheid aan disciplines binnen dit vakgebied zorgt ervoor dat de verwerking van chemisch tekstmateriaal minstens zo complex is als de verwerking van mathematische teksten. Eenvoudige formules als H_2O bevinden zich in dit gebied aan de onderkant, terwijl het andere uiterste gekenmerkt wordt door ondermeer structuren van bijvoorbeeld biochemische verbindingen en complexe organische reactiemechanismen.

Gebrek aan een goed gedefinieerde gebruikersinterface is wel het grootste struikelpunt. Chemische formules kunnen enerzijds via een separaat, al dan niet 'dedicated', tekenpakket worden aangemaakt om vervolgens in teksten opgenomen te worden. Aan de andere kant leveren geïntegreerde pakketten de moeilijkheid *hoe* een gebruiker de formules in een tekst moet opgeven. Een vorm van standaardisatie is helaas nog ver te zoeken.

Toch gaven de afzonderlijke lezingen[1] een duidelijk beeld van de huidige stand van zaken, inclusief de mogelijkheden en onmogelijkheden. Alleen de lezing over TEX bleek, vooral vanwege de persoonlijke voorkeuren van de spreker naast zijn gebrek aan (chemische) LATEX/TEX tekstverwerkingskennis in het algemeen van een niet optimaal niveau.

2 Electronisch publiceren

Een uitstekende keuze van de organisatoren voor wat betreft onderwerp en spreker (P.W.J.M. Boumans) betrof de eerste bijdrage over 'Electronisch Publiceren'. Ondanks het feit dat de lezing nauwelijks iets met *chemische* tekstverwerking te doen had, kwamen toch zaken uitgebreid aan bod die direct te maken hadden met de pipeline 'interactief artikel schrijven' t/m 'het uiteindelijk uitgeven, inclusief de mogelijkheden erna'. Een nieuwe mogelijkheid van electronisch publiceren kwam ter sprake in de vorm van 'Spectrochimica Acta' (waarvan de spreker de 'Editor-in-Chief' is). Deze tijdschriftenreeks heeft nu een echte electronische lijn, t.w. 'Spectrochimica Acta Electronica', welke ingaat op een toekomstige behoefte: het uitgeven van, naast de hardcopy tekst, een diskette welke ondermeer de tekst, graphics, bijbehorende executable programma's, databestanden, parametersets en source code bevat.

Uitgebreid werd door de uiterst boeiende spreker gesproken over de plaats van 'markup'-talen, t.o.v. WYSIWYG en DTP systemen (welke laatste, aldus de spreker, binnen de wetenschappelijke wereld feitelijk al op zijn retour zijn). Ter sprake kwam tevens de plaats van SGML, TEX en LATEX, en de APEM (Auhor's Primary Electronic Manuscript') benadering.
De lezing was een korte uiteenzetting van hetgeen de spreker eerder in een 'standaard werk' had gepubliceerd [2]. Dit laatste is zeker het lezen waard!

[1] De teksten van alle lezingen waren opgenomen in de tijdens de dag beschikbare proceedings; prima verzorgd door de organisatoren!

3 Apple Macintosh omgeving

Tekstverwerking van chemische documenten binnen deze omgeving werd uiteengezet door J. Murre. De mogelijkheden lijken veel op die van de MS-DOS omgeving zoals in de volgende sectie ter sprake komt. Alleen is vanwege de volledige grafische interface van een Apple Macintosh (of te wel de clipboard mogelijkheden), *geen* geïntegreerd tekstverwerkings/teken pakket beschikbaar.

Los van het (algemene) tekstverwerkingsprogramma, zijn er veel *niet-chemische* tekenprogramma's beschikbaar. Voor het direct maken van chemische formules niet altijd geschikt (lees: dus ongeschikt).
Speciale chemische tekenprogramma's die ter sprake kwamen waren ChemDraw, Chemintosh II, ISIS/Draw, Organic Fonts (gebruikt een speciale font set; als font aan te roepen).
De mogelijkheden van deze produkten zijn dikwijls zeer groot, locale intelligentie is aanwezig ten aanzien van 'niet-bestaande' formules, en de kwaliteit is zeer professioneel te noemen, naast de eenvoud in het gebruik, zeker in vergelijking met de hierna beschreven tekenpakketten voor MS-DOS systemen[2]. Niet vreemd eigenlijk voor een Macintosh.

Probleem is echter, en dat geldt ook bij de DOS systemen, dat er geen 'uitwisselbare' (lees: 'zichtbare') markup is gebruikt voor de chemische formules. Het heeft voordelen, doch zeker ook duidelijk veel nadelen!

Algemene gedachten: goede software, leuke hardware (zullen Macintosh bezitters zeker beamen[3]), prima voor locale documenten. Geen standaardisatie. Uitwisseling zeer beperkt, zo niet onmogelijk.

4 MS-DOS omgeving

Een uiteenzetting van chemische tekstverwerking binnen deze omgeving werd gegeven door de heren J.F.M. Bie & W.J.G. Schielen, beiden van het laboratorium Organische Chemie van de Katholieke Universiteit Nijmegen. Ook hier kwam weer de splitsing ter sprake van:

1. **Separate tekstverwerking met een puur chemisch tekenpakket.**
   Als tekstverwerking kan bijvoorbeeld WordPerfect, Word for Windows, Ventura Publisher, Pagemaker (of ook TEX) gebruikt worden. Het tekenpakket kan zijn 'geheel algemeen' (Autocad, Easycad, Harvard-Graphics, SlideWrite of Drawperfect). Niet direct geschikt dus voor chemische formules, veel tijd consumerend, onnauwkeurig, beperkt in gebruik etc. Niet fraai dus.
   Als tekenmodule kan ook een speciaal chemisch tekenpakket genomen worden met als voorbeeld PLT, PsiGen, ChemWindow en Wimp-2001, de laatste twee onder MS-Windows. Alle kennen vanzelfsprekend een set van templates. Bekende eigenschappen zijn: het veel heen en weer switchen tussen tekstverwerking en tekenpakket, transport via HPGL, (E)PS, of CGM file, moeilijk om correcties aan te brengen, soms geen optimale (terug)koppeling etc. Min of meer omslachtig dus; maar het werkt en het biedt voor veel gevallen een oplossing.
2. **Integratie tekstverwerking en tekenpakket.**
   Eerst de eenvoudige programma's (zowel qua tekst als tekenfunctie). Chiwriter, Spellbinder Scientific, Total Word en T-3 kunnen bijvoorbeeld via een speciaal karakterset / 'chemisch font' (niet al te ingewikkelde) chemische formules weergeven. Wel zeer veel beperkingen, zowel wat betreft de chemische formules als zeker voor wat betreft de tekstverwerking zelf.
   Chemtext gaat in mogelijkheden verder, daar het een intern chemisch tekenpakket in zich heeft. Zeer goede tekenmogelijkheden, echter op het gebied van tekstverwerking zijn er toch nog de nodige beperkingen.[4].

Conclusie DOS omgeving: Het blijft behelpen, zeker als men in eerste instantie hoge eisen stelt aan de tekstverwerking zelf. Algemene beperking is ook dat men niet met een *logische markup* werkt, hetgeen de uitwisseling van chemische documenten en het hergebruik binnen andere software modulen zeker niet ten goede komt.

5 TEX presentatie

Wél het enige produkt met een *'visible markup'*. Kenmerkend dus voor onderhoudbaarheid en duidelijke logische structuur, waarbij men gebruik kan maken van de eigen favoriete editor op PC t/m supercomputer. De (logische) markup eigenschappen kwamen in de lezing van P.E.S. Wormer (Theoretiche Chemie, Katholieke Universiteit Nijmegen) echter niet naar voren. Hetzelfde gold ook voor wat betreft de TEX chemische formule mogelijkheid binnen gebieden als anorganische en organische chemie.

De spreker bleek een oude (één van de vroege) gebruikers van TEX te zijn. Hij schroomde ook niet om naast enkele mogelijkheden, vooral de moeilijkheden bij het gebruik van TEX 'overdreven' naar voren te brengen. Veel niet terzake betreffende onderwerpen werden uiteengezet ('Knuth aanhangers zijn soms te vergelijken met sekteleden'[5]).

LATEX zelf had hij echter nooit gebruikt daar hij met TEX

[2] Dat was mijn persoonlijke indruk na diverse demonstraties gezien te hebben.

[3] Voor de duidelijkheid: ik bezit geen Apple Macintosh; ik heb er zelfs nooit mee gewerkt; helaas misschien.

[4] Sorry, ik als LATEX gebruiker ben wat gewend (of verwend??).

[5] Trouwens, en dit komt van mijzelf (GvN), ik las onlangs dat achter de WordPerfect bron de Mormonen uit Salt Lake City schuilgingen. Maar dit is vanzelfsprekend niet ter zake.

alles kon doen. De filosofie achter het TEX/LATEX gebruik werd ook niet genoemd.[6] Daarnaast werden door de spreker onjuistheden weergegeven ('. . . Het nadeel van LATEX op MS-DOS computers is dat de bibliotheek maar net in 640 Kb past. . . ', etc.). Daar LATEX zelf niet in het geheugen van zijn PC paste, had hij het ook nooit gebruikt (. . .). Ik begreep dat hij een gebruiker van emTEX was. Een raadsel dus.
Puur chemische tekstverwerking werd slechts getoond met een (standaard) voorbeeld uit de CHEMSTRUCT macro [5] en het noemen van de LATEX-macro's van Haas & O' Kane [3]. Enige eigen ervaring ontbrak echter bij hem.
Daarnaast kwam ook hier weer naar voren het genereren van chemische formules via een apart tekenpakket (AutoCAD werd daarbij genoemd, en een niet-chemisch voorbeeld ervan getoond. . .). Duidelijk was het dat, mede vanwege zijn chemische studierichting, de spreker nauwelijks tot geen kennis had van *chemische* tekstverwerking.

Jammer eigenlijk deze presentatie. Met een beetje verder zoeken door de organisatoren van dit symposium had (zeker onder de NTG leden) een betere (serieuze én beter bekend met de materie) en objectieve spreker gevonden kunnen worden.

6 Chemische TEX verwerking

Voor wat meer 'volledigheid' en ter verdere studie is het toch zinvol iets te zeggen over de huidige stand van zaken (voor zover mij bekend) op het gebied van 'TEX en chemische tekstverwerking'.
Drie macro pakketten zijn in omloop:

1. Macro's van Roswitha Haas [3]
   Te vinden op enkele ftp sites (inclusief de RUU-server) en op de TEX-SUN distributie tape. Uitgebreide documentatie is beschikbaar. Te gebruiken vanuit zowel TEX als LATEX (gebruikt picture-environment).
2. Macro's van Michael Ramek [5]
   Te vinden op enkele ftp sites. Documentatie geschreven door H. Partl. Ontworpen om gebruikt te worden vanuit de plain TEX omgeving.
3. Macro's van Maurice Laugier [4]
   Te gebruiken in een TEX omgeving.
   Nog niet tegengekomen op een ftp site of een installatie tape (wie wel?). Aan te vragen via `louijean@frgren81`.

Elke set macro's kent zijn eigen goede en minder goede kanten. Ze zijn ontworpen vanuit een organisch chemisch gezichtspunt (tamelijk algemeen dus). Een vergelijking van de diverse macro sets vóór het uiteindelijke gebruik is dus nodig.

Ondanks de dikwijls goede kwaliteit, geldt echter als algemene beperking het gemis van een (gebruiksvriendelijke) gebruikersinterface[7]. Enige oefening en het maken van noodzakelijke aanvullende (eigen) macro's is dikwijls een vereiste om tot een goede oplossing te komen.

7 Conclusie

Een door de KNCV organisatoren uitstekend georganiseerde dag met veel informatie uitwisseling, boeiende demonstraties, een goede set van lezingen en een directe beschikbaarheid van de symposiumlezingen op papier [1]. Alleen jammer dat de TEX lezing hiervan afweek.

Een gestandaardiseerde en geïntegreerde tekst- en chemisch 'teken'pakket is duidelijk missende. Wie voelt zich geroepen?

References

[1] *Molecuul Muis Manuscript* , KNCV Symposiareeks #3, 51 pagina's, (1991).
[2] P.J.M. Boumans, *'The Dissemination of the Results of Scientific Research in the Era of Electronic Media'*, Spectrochimica Acta, Special Supplement, 11–45, (1989).
[3] R.T. Haas & K.C. O'Kane, *'Typesetting Chemical Structure Formulas with the Text Formatter TEX/LATEX'*, Comput. Chem, 11.4, 251–271, (1987).
[4] M. Laugier, *'Composition des formules chimiques en TEX'*, Cahiers GUTenberg #10,11, 209–216, (1991).
[5] M. Ramek, *'CHEMSTRUCT, Chemische Strukturformeln mit TEX'*, Handbuch-Nummer X14, 1–18, (1988).

[6] Degenen die mij een beetje kennen weten dat ik gebruikers ook zoveel mogelijk aanraad om vooral LATEX te gebruiken i.p.v. TEX. Alleen als het niet anders kan moet m.i. naar TEX worden uitgeweken, en dan nog bij voorkeur slechts via (LATEX logische) macro's. Het gebruik van een logische markup staat bij mij nu eenmaal hoog in het vaandel.
[7] Wie staat te trappelen van ongeduld om daar wat aan te doen?

An introduction to TEX

--- part I ---

NTG course June 1992

David Salomon

— 1. Introduction —

TEX is not written TEX, it is not spelled 'T' 'E' 'X', and is not pronounced tex. It is written TEX (a trademark of the AMS), it is spelled Tau Epsilon Chi, and is pronounced tech or rather teck.

TEX is not a word processor. It is a program that sets text in lines and paragraphs—but its design philosophy, its methods, and its approach—are all different from those of a word processor. A modern, state of the art word processor is a WYSIWYG type program; TEX is not, it uses one-dimensional input to generate its two-dimensional output. With a word processor it is easy to generate special printing effects, easy to underline text, to italicize and emphasize; TEX can do those things—and with a lot more attention to detail—but the user has to work harder. The same is true for applications that mix text with graphics. Good, modern word processors can also create diagrams, whereas TEX does not support graphics and some work is necessary to insert diagrams into its output. Other features that are standard and easy to use in a word processor are either non-existent or are hard to use with TEX.

If TEX is not a word processor, then what is it?

From the point of view of the user, TEX is a typesetting program which can be extended to a complete computerized typesetting system by adding a printer driver, the METAFONT program and some utilities.

From the point of view of the computer, however, TEX is a *compiler* whose main task is to *compile* a source program. The source language has variables (with types), block structure, two executable statements (assignment & if), and a powerful macro facility that makes it possible to extend the language. The main output of the compiler is a file containing, not machine instructions, but detailed instructions on how to place characters on a page.

When a compiler sees something in the source file which does not belong in the source language, it issues an error message. TEX, on the other hand, simply typesets any material in the source file that's not part of the language. The source file for TEX contains the text to be typeset, embedded commands, and comments. It's a text file and can thus be prepared with any editor or word processor. TEX generates two outputs, a log file—with run-time information and error messages—and a DVI file (for Device Independent), containing the page coordinates of each character to be typeset. A separate printer-driver program later reads the DVI file and generates printer commands (which are different for different printers) to print the text.

Two font files must exist for each font used in the text. A `.tfm` file and a bitmap file (which is usually called `.gf` or `.pk` file). The `.tfm` file (for TEX Font Metric) contains the dimensions of each character in the font (width, height, & depth) and some other information. The bitmap file contains the actual shape of the characters. TEX only uses the `.tfm` file. When the position of a character on the page is determined, TEX uses the character's width to move the reference point to where the next character should appear. As a result, TEX is ignorant of the actual shape of the characters, and its final output is a list of commands that specify what should be placed on the page and where. A typical command is a triplet of the form:

<char. # in current font, x-coord., y-coord.>

The bitmap file is only used by the printer driver, which sends the individual pixels of each character to the printer.

The diagram above summarizes the relationships between TEX, METAFONT, the printer driver, and the files involved.

To come back to the point of view of the user, the main aim of TEX is to produce *beautifully typeset documents*, especially documents containing mathematics. As a result, TEX is ideal for book publishing, for documents that should look beautiful—such as concert programs and invitations—and for technical publishing. Since typesetting is not a simple process (it is in fact very complex, especially the typesetting of tables and mathematics), TEX is *not easy to use*. It has several features that are complex and hard to master. As a result, new users tend to use TEX only if they need high quality output. When such a user needs a quick letter, a simple memo, or a short note, they usually go back to their familiar word processor. However, advanced users usually have macros that help produce all kinds of documents easily, thus using TEX's power all the time. Either way, if you like beautiful text, or if you write mathematics, TEX is by far the best typesetting tool available today.

To understand the power of TEX, it is necessary to understand the difference between traditional typing and book printing. When upgrading from a typewriter to a computer keyboard, some adjustment is necessary. The keys for '1' and 'l' are identical on a typewriter but different on a computer. Most typewriters only have one, non-oriented single quote ('), but computer keyboards typically have two oriented single quotes, a left (‘) and a right (’). When taking the next step of upgrading from word processing to typesetting (or book printing), a few more adjustments become necessary. The most important ones are:

▪ Computer keyboards have just one kind of double-quote (") but books have two kinds, a left double-quote (“) and a right one (”). To produce them in TEX, you simply type two single quotes of the apropriate kind in a row. Thus to typeset “with regret.” you need to type `‘‘with␣regret.’’`

▪ Another important difference is the dash (or hyphen). Computer keyboards have only one dash, namely (-); in a carefully typeset book, however, there are four different symbols:

a hyphen (-);
an en-dash (–);
an em-dash (—);
a minus sign ($-$);

Hyphens are used for compound words like 'right-quote' or 'X-Register'. En-dashes are used for number ranges, like 'Exercises 12–34'. They are obtained by typing two consecutive dashes '--'. Em-dashes are used for punctuation in sentences—like the ones around this section—they are what we usually call dashes. To get one, just type three consecutive dashes. Minus signs are, of course, used in typesetting mathematics.

▪ The main task of TEX can be described as converting a one-dimensional stream of text into a two-dimensional page where all lines have the same width. This is done by stretching or shrinking the spaces between words on each line. In fine typesetting, however, those interword spaces have limited flexibility, so sometimes words have to be hyphenated. TEX uses a sophisticated hyphenation algorithm, so hyphenation is automatic. In cases where the algorithm does not work, the user can specify the correct hyphenation of words.

▪ Well-printed books use ligatures and kerning. Certain combinations of letters, like 'ff', 'fl', 'fi', 'ffl', look better in the traditional roman type when the letters are combined. Such a combination is called a ligature. Foreign languages may have other ligatures, such as 'ij' in Dutch. Compare, for instance, the word 'fluffier' set by TEX to the same word 'fluffier' generated by a word processor. When a font is designed for TEX, the designer should specify all the special combinations that should be replaced by ligatures, and **design** the ligatures. That information goes into the font's `.tfm` file and is used by TEX to substitute the ligatures automatically. TEX can even remove a ligature later, for example, if it decides to hyphenate the word at that point.

Kerning refers to certain letter combinations that should be moved closer (negative kerning) or apart (positive kerning) for better appearance. An 'A' adjacent to a 'V' is a good example (compare 'AVAV' to 'AVAV'). Other examples are 'away', 'by', 'ox', 'ov', 'xe', and 'OO' (the last one has positive kerning). Again, the font designer decides what the kerning should be, and that information also goes into the font's `.tfm` file. TEX, of course, allows the user to easily override ligatures and to change the kerning to any desired value.

Ligatures and kerning improve the relationship between adjacent letters. They, together with hyphens and flexible interword spaces, are the main participants in the delicate balancing act required for the line break decisions.

Considering the power and flexibility of TEX, it is surprising that its basic algorithms are based mostly on three concepts—*boxes, glue* and *penalties*.

- A box in TEX is an indivisible unit of material to be typeset. TEX will not break the contents of a box across lines or between pages.

To begin with, each character in a font is enclosed in a box whose dimensions (width, height, and depth) become the dimensions of the character. When TEX typesets a word, it pastes the individual character boxes side by side, with no spaces in between (except when kerning demands shifting boxes horizontally, or when the user wants boxes moved—which is how the TEX logo is produced). The result is a box containing the typeset word. The width of such a box is the total widths of the boxes inside, plus the kernings which, of course, may be negative. The height of the word-box is the maximum height of the component boxes, and the same is true for the depth.

Similarly, when TEX decides to break a line, it pastes the individual word boxes side by side, with appropriate spaces between them, and generates a new horizontal box, a line-box. The width of the line-box is the sum of widths of the word boxes inside it plus the sum of the spaces (glue) between the individual word boxes. The height and depth of the line box are the maximum heights and depths of the component boxes.

To set an entire page, TEX accumulates enough horizontal line-boxes; it then pastes them vertically, one below the other, with appropriate interline spaces, to generate a vertical page-box. The next step is to call an *output routine* which is either written by the user or supplied by TEX. The output routine adds finishing touches such as heading, footing, and page numbers. It may even decide to trim the page, to return part of the bottom of the page to TEX (to become the top of the next page), or do anything else to the page. The output routine should normally execute `\shipout`, which translates the contents of the page box to DVI file specs. TEX then continues to read the source to build the next page.

- The term 'glue' refers to the spaces between boxes. In order to justify the text, TEX adjusts the spaces between words (but not the spaces between characters in a word). Those spaces must, therefore, be flexible. The same applies to interline spaces on the page. A glob of glue in TEX is a triplet $< w, y, z >$ where w specifies the natural size of the glue, y is the amount of stretch in the glue and z, the amount by which it can shrink. For interword glue, these values depend on the font and are specified by the font designer. For the standard 10 point roman font used by TEX, the values are <3.3333pt, 1.66666pt, 1.1111pt>. The line breaking algorithm considers various alternatives of combining words into lines, and for each alternative it calculates the amount by which the individual globs of glue have to be stretched or shrunk. After considering all the possible line breaks for an entire paragraph, the best line breaks are chosen (best in a sense that will be explained elsewhere), the entire paragraph is set, and TEX reads the next item from the source file (usually the start of the next paragraph).

- The third mathematical construct used by TEX is the penalty. A penalty can be inserted into the text at any point either explicitly, by the user, or automatically, by TEX. The penalty specifies the desirability of breaking a line or a page at the point, and it is used to discourage bad breaks, to encourage good ones, to force certain breaks, and to avoid others. Penalty values are in the range $[-10000, 10000]$. Any value ≤ -10000 is treated as $-\infty$, and any value ≥ 10000 is considered $+\infty$. The user may, for example, insert '`\penalty100`' at a certain point in a paragraph, which has the effect that, if TEX decides to break the line at that point, a penalty of 100 will be added to the line, making that breakpoint less desirable. This tends to discourage line breaking at that point. A negative penalty is actually a merit, and it encourages a break. Infinite penalty prohibits a line break where it is specified, and $-\infty$ forces one. The commands (control sequences) `\break` `\nobreak` can be used to insert those infinite penalties at any point in the text. There are two common examples of penalties. The first occurs at any hyphenation. When TEX decides to hyphenate a word, it inserts a penalty of 50 at any potential hyphenation point. The second has to do with psychologically bad breaks. In a text containing '...Appendix G' it is psychologically bad to break between the two words because it interrupts the smooth flow of reading. To prevent such a break, a TEX user should type '`Appendix~G`'.

The tilde (~) acts as a 'tie'. It ties the two words such that in the final document there will be a space between them but no line break.

Similar examples are:

Table~G-5	Figure~18	dimension~d	Louis~XVI	1,~2, or~3	from 0 to~1
Y~Register	modulo~e^x	Rev.~Henry	HRH~prince Abdul		

Each tie is converted by TEX into a penalty of $-\infty$ to prevent a break.

This mechanism of boxes, glue, and penalties to arrange text in lines and pages, has proved extremely flexible and powerful. It makes it possible to set text in non-standard ways to achieve special effects. Features such as narrow paragraphs, newspaper formats, paragraphs with variable line widths, punctuations in the margins, ragged right margins, centered text, and complex indentations, can all be achieved with TEX. This mechanism is one of the main innovations introduced by TEX.

— 2. Line breaking and page layout —

To achieve a straight right margin, TEX adjusts the spaces between words but not the spaces between characters in a word. The boxes defining each character are juxtaposed with no intervening spaces, for the following reasons:

- It makes for a more uniform appearance of the page. Because of the nature of human vision, spaces between words disturb the eye less than spaces between characters in a word.
- In an underlined font, spacing the characters would break the underline into separate segments.

This is perhaps a good point to discuss underlining in TEX. Underlining is one of the features that distinguish TEX from word processors, and can give an insight into the design philosophy of TEX. In a word processor it is usually easy to underline text; in TEX, underlining is discouraged. The reason is that TEX is not a word processor but a typesetter designed to produce beautiful books. In a book, underlining is rare, and emphasizing is done using either **boldface**, *italics*, or *slanted* fonts. If a certain text requires a lot of underlining, the best way to do it in TEX is to design a special font in which all the letters are underlined. Such a font requires no spaces between the characters in a word.

To achieve a uniform page layout, TEX uses two principles: 1. The vertical distance between lines is kept as constant as possible, using the rules below. 2. When a good page break requires squeezing another line on a page, or removing a line from a page, TEX changes the vertical distance between **paragraphs** (or between math formulas), not between lines. This again has to do with the way our eyes see, and guarantees that all the pages would appear to have the same proportion of black to white areas.

To determine the vertical distance between consecutive lines, three parameters a, b, c are used. Their values depend on the font, and for the common 10 point roman font they are: $a = 12$pt, $b = 0$, $c = 1$pt. Typically, consecutive lines are not juxtaposed vertically but are spaced such as to make the distance between consecutive baselines equal to a. However, if the distance [top of lower line]$-$[bottom of upper line] is less than b, then the lines are spaced such that that distance is set equal to c. This may happen if the line contains a large (say, `18pt`) character. Parameters a, c are of type `<glue>` so they may have flexibility. Typically this flexibility is zero, but the user may want to define, e.g. $a =< 12, 2, 1 >$. Such value makes sense for a short, one page, document. The flexibilty of the glue would make it easier for TEX to fit the text on one page.

Any discussion of TEX's line breaking algorithm should start with an outline of a typical line breaking algorithm used by a modern, commercial word processor. The method uses three values—for the natural, minimum and maximum spacings between words—and proceeds by appending words to the current line, assuming natural spacing. If, at the end of a certain word, the line is too long, the algorithm tries to shrink the line. If that is successful, the next word will start the next line, and the current line is printed. Otherwise, the word processor discards the last word and tries to stretch the line. If that is successful, the discarded word becomes the first one of the next line. If neither shrinking nor stretching works (both exceed the preset parameters), a good word processor tries to hyphenate the offending word, placing as much of it on the current line as would fit. The user may be asked to confirm the hyphenation, and the rest of the hyphenated word is placed at the start of the next line. A word processor that does not hyphenate has to resort to overstretching and may generate very loose lines.

The important feature of all such methods is that once a break point is determined, the algorithm does not memorize it but starts afresh with the next line. We can call such an algorithm 'first fit' and its main feature is that it does not look at the paragraph as a whole. Such an algorithm produces reasonably good results. For a high quality typesetting job, however, it is not fully satisfactory. For such a job, an algorithm is needed which considers the paragraph as a whole. Such an algorithm makes only tentative decisions for line breaks and may, if something goes bad toward the end of the paragraph, go back to the first lines and change their original, tentative, breakpoints. TEX's algorithm determines several feasible breakpoints for each line and calculates quantities called the *badness* and *demerits* of the line for each such breakpoint. After doing this for the entire paragraph, the final breakpoints are determined in a way that minimizes the demerits of the entire paragraph. Mathematically the problem is to find the shortest path in an acyclic graph.

— 3. Fonts —

Traditionally, the word *font* refers to a set of characters of type that are all of the same size and style, e.g., Times Roman 12 point. A *typeface* is a set of fonts of different sizes but in the same style, e.g., Times Roman. A *typeface family* is a set of typefaces in the same style, e.g., Times.

The size of a font is normally measured in points (more accurately *printer's points*), where 72.27 points equal 1 inch. The reader should refer to [Ch. 10] for a description of all the valid dimensions in TEX. The *style* of a font describes its appearance. Traditional styles are roman, **boldface**, *italic*, *slanted*, `typewriter` and ßsans serif. In TEX, a font can have up to 256 characters, although most fonts only have 128 characters.

— 4. The CM fonts —

Computer Modern (CM) is a metafont, developed in METAFONT, from which many different fonts have been derived, by different settings of parameters. The fonts all look different, but they blend together. They are called the CM fonts, and their names start with 'cm'. The standard CM fonts are:

- `cmr` or Roman. These are used for plain text. The standard sizes are (the '*' indicates fonts that are automatically loaded by the `plain` format) `cmr17 cmr12 cmr10* cmr9 cmr8 cmr7* cmr6 cmr5*`.
- `cmsl` or Slanted. They are slanted versions of the `cmr` characters. The standard sizes are `cmsl12 cmsl10* cmsl9 cmsl8`.
- `cmdunh` or Dunhill. Same as `cmr` but with higher ascenders. Only `cmdunh10` is standard.
- `cmbx` or Bold Extended. These are used for Boldface characters. `cmbx12 cmbx10* cmbx9 cmbx8 cmbx7* cmbx6 cmbx5*`.
- `cmb` or Bold. This is bold but too narrow for normal use. `cmb10`.
- `cmbxsl` or Bold Extended Slanted. `cmbxsl10`.
- `cmtt` or Typewriter. Fixed-space, resembling old typewriter style. `cmtt12 cmtt10* cmtt9 cmtt8`.
- `cmvtt` or Variable Typewriter. Same as `cmtt` but with proportional spacing. `cmvtt10`.
- `cmsltt` or Slanted Typewriter. `cmsltt10`.
- `cmss` or Sans Serif. Used for headings and for formal texts. `cmss17 cmss12 cmss10 cmss9 cmss8`.
- `cmssi` or Sans Serif Italics. `cmssi17 cmssi12 cmssi10 cmssi9 cmssi8`.
- `cmssbx` or SS Bold Extended. `cmssbx10`.
- `cmssdc` or SS Demibold Condensed. Normally used for chapter headings. `cmssdc10`.
- `cmssq` or SS Quote. Special SS font for quotations. `cmssq8`.
- `cmssqi` or SS Quote Italics. Again used for quotations. `cmssqi8`.
- `cminch` or Roman Inch. These are used for titles. `cminch`.
- `cmfib` or Fibonacci. In this version the parameters have relative sizes determined by the Fibonacci sequence. `cmfib8`.
- `cmff` or Funny Font. Different from the other cm fonts. Rarely used. `cmff10`.
- `cmti` or Text Italics. This is the normal italics font. `cmti12 cmti10* cmti9 cmti8 cmti7`.
- `cmmi` or Math Italics. Slightly different from text italics, and without spaces (spaces are automatically supplied in math mode). `cmmi12 cmmi10* cmmi9 cmmi8 cmmi7* cmmi6 cmmi5*`.
- `cmbxti` or Bold Extended Text Italics. `cmbxti10`.
- `cmmib` or Math Italics Bold. For math mode. `cmmib10`.
- `cmitt` or (text) Italics typewriter. `cmitt10`.
- `cmu` or Unslanted (text) Italics. Unslanted version of cmti. Used for Editor's notes in TUGboat. `cmu10`.
- `cmfi` or Funny Italics. An Italics version of cmff. `cmfi10`.
- `cmsy` or Math Symbols. Contains the math symbols normally used by TEX. `cmsy10* cmsy9 cmsy8 cmsy7* cmsy6 cmsy5*`.
- `cmbsy`. A Bold version of the math symbols. `cmbsy10`.
- `cmex` or Extension. More math symbols. `cmex10*`.

- `cmtex` or TEX Extended. An extended ASCII font. `cmtex10 cmtex9 cmtex8`.
- `cmcsc` or Caps & Small Caps. Contains small caps instead of lower case letters. `cmcsc10`.
- `cmtcsc`. A Typewriter version of `cmcsc`. `cmtcsc10`.
- `cmc` or Concrete. These were specially developed for the book *Concrete Mathematics*, to blend with the Euler math fonts.

Some of these fonts are rarely used, but were easy to obtain, by trying various settings of parameters. Any special sizes not mentioned above should also be easy to derive.

In `plain` TEX, the default font is cmr10. Also, macros `\bf`, `\it`, `\sl` and `\tt` are set [351] to select the different styles in 10 point size. If large parts of the document should be typeset in, say, 12 point, the definitions of `\bf` and its relatives should be changed accordingly. To do this, the different twelve point fonts should be loaded, at the start of the document, and assigned names by

```
\font\twerm=cmr12
\font\twebf=cmbx12
\font\tweit=cmit12
\font\twesl=cmsl12
\font\twett=cmtt12
```

Macro `\twelve` should then be defined as

```
\def\twelve{\def\rm{\twerm}\def\bf{\twebf}\def\it{\tweit}%
\def\sl{\twesl}\def\tt{\twett}%
\rm}
```

Text areas that should be set in 12 points should be bounded by `\begingroup\twelve` and `\endgroup`.

The CM family (Ref. 1) represents the most ambitious attempt so far to develop a general metafont. It is based on an earlier version called AM (almost modern), which is now obsolete. Two interesting adaptations of CM are outline fonts (Ref. 2) and the Pica fonts. The reader should refer to [Chs. 2, 4, 9 & App. F] for more information on the CM fonts.

Many special fonts have been developed in METAFONT. Examples are exotic languages, music notes, chess figures, astronomical symbols & logic gates. Ref. 3 is a detailed listing. The MetaFoundry (Ref. 4) developed many fonts in the early 1980s. However, very few other metafonts exist, the most well known of which are:

- Pandora, developed by N. Billawala (Ref. 5).

- The Euler family, designed by Herman Zapf, and developed at Stanford (Ref. 6). It is not a true metafont, as the characters were digitized.

- The Gothic family, including Fractur and Schwabacher, developed by Y. Haralambus (Ref. 7).

— 5. Font examples —

This is an example of font cmr10 (roman)

This is an example of font cmbx10 (bold extended)

Some math symbols $\bigcap \big[\big] \Big\surd \big] \Big\surd \coprod \big\{ \widetilde{} \uparrow \coprod \big| \surd \big\lceil \widetilde{}$

Notice the absence of spacing in the next two lines. Math italics is automatically used and spaced in the math mode.

$Thisisanexampleoffontcmmi10\leftarrow mathitalics\leftarrow$

$\scriptscriptstyle Thisisanexampleoffontcmmi5\leftarrow mathit5pt\leftarrow$

This is an example of font cmti10 (text italics)

`This is an example of font cmtt10 (typewriter)`

This is an example of font helvetica (sans serif)

`This is an example of font Courier (fixed spacing)`

This is an example of font Palatino

This is an example of font times (roman)

THIS IS AN EXAMPLE OF FONT CMCSC10 (CAPS AND SMALL CAPS)

This is an example of font cmdunh10 (ascenders).

This is an example of font cmssdc10 (sans serif demibold condensed)

— 6. Magnification —

The \magnification command scales the entire document. Everything—except the width & height of the text, and the margins—is made bigger or smaller. The magnification gactor in an integer, thus '\magnification=2000' scales everything by a factor of 2 (the document will spread over more pages), and '\magnification=500' makes everything half its original size. Note that this command can only be used once in a document, and should be placed at the beginning.

It is also possible to magnify individual fonts using the scaled parameter. Thus if font cmr12 is not available, it is possible to say '\font\twelve=cmr10 scaled 1200'. However, font \twelve will not look as good as the real thing.

Experience shows that certain font sizes blend together better than others. To encourage users to use those sizes, plain TEX includes [349] the quantities \magstep0 (= 1000), \magstep1 (= 1440), \magstep2 (= 1728), \magstep3 (= 2074), \magstep4 (= 2488), and \magstephalf (= 1095). Again, it should be emphasized that magnification of fonts reduces their quality. For best results, all the fonts needed in a document should be available without having to magnify anything.

— 7. Advanced Introduction —

The rest of this chapter is an advanced introduction to TEX, stressing the main parts, main operations, and certain, advanced, concepts. As with most other presentations of this type, some terms have to be mentioned before they are fully introduced, so the best way to benefit from this material is to read it twice.

— 8. Registers —

A register is temporary storage, a place where data can be saved for later use. Using a register thus involves two steps. In the first step something is stored in the register; in the second step, the contents of the register is used. With the exception of box registers, the contents of a register can be used and reused indefinitely. There are six classes of registers, summarized in the following table.

Class	Contents	Default value
\count	integer	0
\dimen	dimension	0pt
\skip	glue	0pt plus0pt minus0pt
\muskip	muglue	0mu plus0mu minus0mu
\toks	token string	empty
\box	a box	void

Note that each class of registers can only be used for data of a certain type. The registers are thus similar to strongly typed variables used in many programming languages. Each class contain 256 registers, so there are, e.g., the 256 count registers \count0 through \count255.

Reserved Registers. Certain registers are reserved by TEX for special purposes. \box255 is used by the OTR. \count0 through \count9 are used by the plain format for the page number. Those registers should not be used unless you know what you are doing.

Declaring registers. Since it is not a good idea to refer to registers explicitly, the plain macro \new should be used to declare registers and assign them names. Thus \newcount\temp is a typical use. It allocates the next available count register, and assigns it the name \temp.

Five of the six register classes are similar, but box registers are different. Boxes have dimensions and internal structure, they can be nested by other boxes, and tend to consume memory space. As a result, there are differences between box registers and other registers.

- The command \newcount\temp creates a quantity \temp whose value is, e.g., \count18. In contrast, \newbox\Temp creates \Temp as the number 18. Saying 'AıB' is thus identical to 'A` B'. (This can be verified by \show\temp.)

- Assigning a new value to a register is done by an assignment statement of the form \temp=1234. Assigning a new value to a box register is done by means of '\setbox\Temp=...'.

- Box register 255 is reserved for the use of the OTR. Registers `\count255`, `\skip255` etc. are available for general use.

- A box register is emptied when it is used. The register thus becomes void. In contrast, other registers cannot be void; they must always contain a value.

- The contents of any register, except a box register, can be written on a file. Thus we can say, e.g., `\write\abc{\the\skip0}`, but we cannot say `\write\abc{\the\box0}`. The reason is that boxes are complex structures.

- The primitive `\the` can be used to produce the value of any register except a box register.

- At the start of a job boxes are normally void, but there is a subtle point, involving `\box0` and the `plain` format, that users should keep in mind. The `plain` format says (on [361]) `\setbox0=\hbox{\tenex B}`. This is used in the definition of macro `\bordermatrix`. As a result, `\box0` is not void at the start of a job. This point is mentioned again, in connection with `\lastbox`.

— 9. `\the` —

The primitive `\the` can be used to produce the values of certain internal quantities. TEX novices are always confused by it. A typical complaint is "Why do I say '`\box0`' to typeset `\box0` but '`\the\count0`' to typeset `\count0`?" The answer is—to make it easier for TEX to compile the document and to detect errors.

The command `\the` must be followed by an internal quantity, such as a register. It produces tokens that represent the value of the quantity. Thus after saying '`\newcount\temp \temp=1234`', the command `\the\temp` creates the tokens '1234'. Thus '`\hskip\the\temp pt`' will skip 1234 points, and '`ABC\the\temp GHK`' will typeset '1234' between the `C` and the `G`.

The command `\temp`, in contrast, does not create tokens. TEX considers it the start of an assignment, unless it expects an integer at this point. Consider, e.g., the command '`\hskip\temp pt`'. After TEX has read the `\hskip`, it expects a dimension (a number followed by a valid unit of dimension). This command will thus skip 1234 points.

The two examples above suggest that `\temp` and `\the\temp` produce the same results, but this is not generally true. Consider the assignment '`\skip0=3pt plus 2pt`'. Saying '`\hskip\the\skip0 minus 1pt`' will skip by '`3pt plus 2pt minus 1pt`', but saying '`\hskip\skip0 minus 1pt`' will skip by '`3pt plus 2pt`' and will consider the '`minus1pt`' text to be typeset.

In the former case, the tokens produced by `\the\skip0` blend with the rest of the document and become part of the `\hskip` command. In the latter case, TEX expects the `\hskip` to be followed by a valid dimension and, on finding `\skip0`, is satisfied and executes the command, not looking for any more arguments.

— 10. Modes —

Of all the advanced concepts, the idea of modes [Ch. 13] is, perhaps, the most important. At any given time, TEX is in one of six modes, and its behaviour depends on the current mode. The mode can frequently change and can also be nested. The six modes are:

- Horizontal, or H, mode. TEX is in this mode when it reads the text of a paragraph.

- Vertical, or V, mode. This is where TEX usually spends its time between paragraphs, executing commands.

- Restricted horizontal (or RH) mode. TEX switches to this mode when it builds an `\hbox`. This mode is very similar, but not identical to, H mode.

- Internal vertical (or IV) mode. Commands such as `\vbox` or `\vtop` force TEX to go into this mode, which is very similar, but not identical to, V mode.

- Inline math mode, where a math inline formula is built.

- Display math mode, where a display formula is constructed.

When TEX starts, it is in V mode (between paragraphs). It reads the input file and, when it sees the first character to be typeset (or anything that's horizontal in nature, such as `\noindent`, `\vrule`), it switches to H mode. In this mode, it first executes the tokens in `\everypar`, then reads the entire paragraph into memory. The paragraph is terminated by `\par`, by a blank line, or by anything that doesn't belong in H mode (such as `\vskip` or `\hrule`). TEX then switches to V mode, where it sets the paragraph and takes care of page breaking.

A paragraph is set by breaking it into lines which are appended—each as an `\hbox`—to the *main vertical list* (MVL). After appending the lines of a paragraph to the MVL, TEX determines whether there is enough material in the MVL for a full page. If there is, the page breaking algorithm is invoked to decide where to break the page. It moves the page material from the main vertical list to `\box255` and invokes the output routine. Some material is usually left in the MVL, to eventually appear at the top of the next page. The routine can further modify `\box255`, can add material to it or return some material from it to the main vertical list. Eventually, the output routine should invoke `\shipout` to prepare the DVI file. TEX stays in V mode and continues reading the input file.

Modes can be nested inside one another. When in V mode, between paragraphs, TEX may be asked to build a `\vbox`, so it enters IV mode. While in this mode, it may find characters of text, which send it temporarily to H mode. In that mode, it may read a '`$`', which causes it to switch to inline math mode. The curious example on [88] manages to nest all the modes at once.

— 11. Anatomy of TEX —

A quote, from [38], is in order. "It is convenient to learn the concept (of tokens) by thinking of TEX as if it were a living organism." We will develop this concept further, and try to get a better understanding of the overall organization of TEX by considering the functions performed by its main "organs", namely eyes, mouth, gullet, stomach, and intestines (see anatomical diagram on [456]).

▪ TEX uses its "eyes" to read characters from the current input file. The `\input` command causes TEX to 'shift its gaze' and start reading another input file.

▪ In its "mouth", TEX translates the input characters into tokens, which are then passed to the "gullet." Spaces and end-of-line characters are also converted into tokens and sent to the gullet. A token is either a character of text or a control sequence. Thus the name of a control sequence, which may be long, becomes a single token. The process of creating tokens involves attaching a category code (see below) to each character token, but not to a control sequence token.

▪ The "gullet" is the place where tokens are expanded and certain commands executed. Expandable tokens [373] are macros, `\if...\fi` tests, and some special operations such as `\the` and `\input`.

A token arriving at the gullet is sent to the stomach, unless it is expandable. Expanding a token results in other tokens that are, in turn, either expanded, if they are expandable, or sent to the stomach. This process continues until no more expandable tokens remain in the gullet, at which point the next token is moved from the mouth to the gullet, starting the same process (the word "regurgitation", on [267], nicely describes this process). Certain commands, such as `\expandafter` & `\noexpand`, affect the expansion of tokens, so they are executed in the gullet.

Expandable tokens are macros, active characters, conditionals, and some primitives, e.g. `\romannumeral`, listed on [215]. They are expanded in the gullet. For macros with no parameteres the expansion is a simple replacement (also for some primitives, such as `\jobname`). Normally, however, the token expanded depends on arguments, which have to be read before the expansion can take place.

Exceptions: The construct `\expandafter`$\langle\text{token}_1\rangle\langle\text{token}_2\rangle$ is treated by the gullet in a special way. It is replaced by $\langle\text{token}_1\rangle\langle\text{expansion of token}_2\rangle$, which is then scanned again by the gullet.

A `\noexpand`$\langle\text{token}\rangle$ prevents the gullet from expanding the token.

▪ As a result, there is a constant stream of tokens arriving at the "stomach", where they get executed. Most tokens are characters of text, and are simply typeset (appended to the current list). Tokens which are TEX primitive commands are executed, except that the "stomach" may have to wait for the arguments of the command to arrive from the gullet. Recall that non-primitive commands are macros and are expanded in the gullet.

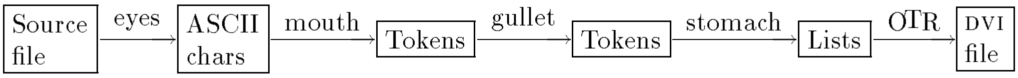
Another way of explaining stomach operations is: The stomach executes tokens coming from the gullet. It classifies all tokens into two groups, tokens used to construct lists, and tokens that are mode independent. The former group includes characters of text, boxes and glue. They are mode sensitive, can change the mode, and are appended to various lists. The latter can be assignments, such as `\def`, or other tokens, such as `\message` or `\relax`. The `\relax` control sequence deserves special mention. It is a primitive and thus unexpandable. The gullet passes it to the stomach, where its execution is trivial. It is an important control sequence, however, because it serves as a delimiter in both the gullet and the stomach.

The result of executing tokens in the stomach is larger and larger units of text. Individual characters are combined to make words, which are combined to form lines, which are combined to make pages of text. When the next page of text is ready, it is sent to the "intestines."

■ The output routine corresponds to the "intestines" of TeX. It receives a page of text and, after some processing, translates it into DVI commands that are appended to the DVI file. The processing may consist of adding something to the page (such as a header, a footer, footnotes or margin notes) or deleting something from it (certain lines or even a big chunk). The deleted material is either discarded or is returned to the stomach.

The DVI file is the final output of TeX, and it comes out of the intestines in pages. It consists of commands that tell where each character is to be placed on the page. Ref. 8 is a detailed description of the DVI file format.

The entire anatomy is summarized in the diagram below.



The advantage of this anatomical description is that we can think of the process as a pipeline. Material (mostly tokens) advances from organ to organ and is processed in stages. However, the individual organs sometimes affect each other. The best example is the `\catcode` primitive. It is executed in the stomach, and immediately starts affecting the way the mouth assigns catcodes to tokens. Another example is the `\def` primitive. When a `\def\abc` is executed, the definition becomes available for the gullet to use whenever `\abc` has to be expanded. As soon as another `\def\abc` is executed in the stomach, the gullet is affected, and will use the new definition in future expansions of `\abc`.

— 12. Characters —

TeX inputs characters from the input file and outputs characters to the DVI file, so characters are important for an overall understanding of TeX. Characters usually come from the input file (through the eyes), but may also come from the expansion of macros (in the gullet). These are the only character sources of TeX. Most characters are simply typeset, but some, such as '\' and '`$`', have special meanings and start TeX on special tasks.

A character is input as an ASCII code, which becomes the *character code*. The character then gets a *category code* attached to it (in the mouth), which determines how TeX will process the character. When a macro is defined (in the stomach), catcodes are attached to each character (actually, to each character token) in the definition, and are used when the macro is later expanded (in the gullet). Advanced users would like to know that a certain amount of processing goes on even in the mouth. Among other things (see complete description on [46–49]) consecutive spaces are compressed into one space, carriage returns are inserted at the end of each line, consecutive spaces at the beginning of a line are ignored, and spaces following a control word are ignored (they never become space tokens).

There is a simple way to find the ASCII code of a character. Just write a left quote followed by the character [44]. Thus `` `b `` has the value 98, and `` \number`b `` will actually typeset a 98.

Exercise 1: What will be typeset by `` \number`1 `` and what, by `` \number`12 ``?

Answer. The result of `` \number`1 `` is 49, the character code of '1', being typeset. The result of `` \number`12 `` is the same 49, which is typeset, and is immediately followed by a 2; thus 492.

Exercise 2: What is the result of `` \number`{0} `` and what, of `` \number`% ``?

Answer. The character code of '`{`' (123) will be typeset, The right brace will cause an error (too many '`}`'). The `%` in `` \number`% `` will be considered a comment, so the result will be the code of whatever character happens to follow (see later on how to get the character code of `%`).

The ASCII table provides 128 codes, but keyboards normally don't have that many keys. The `\char` command or the '`^^`' notation [45–46] can be used to refer to a keyless character. Thus the notation '`^^`x', where x is any character, refers to the character whose ASCII code is either 64 greater than, or 64 less than, the ASCII code of x. Examples are '`^^M`' (return), '`^^J`' (line feed), '`^^@`' (null), '`^^I`' (horizontal tab), and the other ASCII control characters. The `\char` command is easy to use; if the current font is cmr10, then `\char98` is the code of 'b'. In general, `\char127` is the code of the character in position 127 of the current font.

The difference between the two notations is that `\char` is easier to use but is executed in the stomach (i.e., late); in contrast, a ‘`^^`*x*’ is preprocessed into a single token in the mouth (i.e., as soon as it is input). As a result, the concoction `\def\a^^"c{...}` defines a macro `\abc`, but `\def\a\char98c{...}` defines a macro `\a` with the string `\char98c` as a delimiter.

— 13. End of line —

We intuitively think of a line of text as ending with a carriage return. TEX, as usual, offers a more general treatment of this feature. At the end if every line of text, a special character is inserted, that is the value of parameter `\endlinechar` [48]. This parameter is set by `INITEX` to a ⟨return⟩ (`^^M`), but can be changed by the user. If `\endlinechar` is negative or is greater than 255, no end-of-line character is inserted. In this case, the input is considered one long line. The same effect can be obtained by ending every line with a comment character.

— 14. Numbers —

Not everything can be typeset by `\the`. The definition of a macro (its replacement text), e.g., can only be typset by `\meaning`. To print a numeric quantity, the primitive `\number` can always be used. Even if the number is the value of a macro. Defining the macro `\def\ctst{10}`, we cannot say `\the\ctst`, but we can say `\number\ctst`. Also `\dimen` registers are printed differently by `\the` and by `\number`.

The left-quote character usually acts as a normal character of text. Sometimes, however, it signifies the start of a number. Similarly, the right-quote and double-quote characters sometimes have special meanings; they signify the start of an octal or hexadecimal number. When these characters arrive at the stomach, they are tokens with catcode 12 (other). If the stomach is expecting a number, it will assign them their special meanings and expect them to be followed by digits (decimal, octal, or hex). Otherwise, they will simply be typeset.

The following are all valid representations of the decimal number 98: `98 +98 098 '142 "62` `` `b `\b ``. They may be used with any command requiring an argument of type ⟨number⟩ (see definition of ⟨number⟩ on [269]). Thus `` \number`b `` wil typeset 98, and `\catcode+98="D` will convert the letter ‘b’ into an active character. (Incidentally, once you do that, ‘b’ isn’t a letter any longer, so something like `\bye` will be interpreted as the control symbol `\b` [52], followed by the letters ‘ye’.)

Exercise 3: What is the result of `` \catcode101=14 \number`e ``?

Answer. The character code of ‘e’ is 101, so the assignment makes ‘e’ a comment character. The `\number` is now considered the control sequence `\numb` (which is normally undefined), followed by a comment. The result is the error message `! Undefined control sequence \numb`.

Both notations `` `b `` and `` `\b `` produce the character code of ‘b’. The difference between them is that the latter form can be used with any character. Thus `` \number`\% `` produces 37 but `` \number`% `` will treat the ‘%’ as a comment, and will look for an argument on the next line. Similarly, `` \number`\^^M `` produces 13 (ASCII ⟨return⟩), but `` \number`^^M `` will consider the `^^M` an end-of-line, will replace it with a space, and will produce 32 (ASCII ⟨space⟩).

Non-integer numbers can only be used with dimensions, and are considered multipliers. Thus `1pt` multiplies the basic unit of a `pt` by one, and `2.5\baselineskip` mutiplies the current natural size of `\baselineskip` (not its stretch and shrink components) by 2.5.

— 15. Category Codes —

One of the main considerations behind the design of TEX was to make it as general and flexible as possible, so it could be adapted to many different tasks. Thus the character ‘\’ is normally used to start the name of a control sequence (it is the *escape character*), but if the ‘\’ is needed for other purposes, any character can be defined as the escape character. The same thing is true for the other special characters, namely { } $ & # ^ _ and %.

Each of those characters is special only because it is assigned a special *category code* when TEX starts. It is easy to change the category codes, thereby changing the meanings of characters. There are 16 category codes [37] numbered 0 to 15:

0	Escape character	4	Alignment tab	8	Subscript	12	Other character
1	Begin. of group	5	End of line	9	Ignored character	13	Active character
2	End of group	6	Parameter	10	Space	14	Comment character
3	Math shift	7	Superscript	11	Letter	15	Invalid character

The category code of the character ‘A’ can be typeset by the command \the\catcode‘A. It can be displayed in the log file by \showthe\catcode‘A. The category code is assigned to a character in the mouth, and that assignment is permanent. The pair ⟨character code, category code⟩ becomes a *token*. The mouth also converts control sequences into tokens, but they do not get a catcode.

When a macro is defined (in the stomach) each token of the replacement code gets a catcode assigned. When the macro is later expanded (in the gullet), the replacement code is copied, with arguments replacing the parameters, and the resulting tokens are sent to the stomach.

The catcode of a character can be changed by the \catcode primitive. The most common example is \catcode‘\@=11, which makes ‘@’ a letter. It can now be used in control words, as any other letter, and the plain format [App. B] makes heavy use of this in order to define ‘private’ macros, inaccessible to the user. Another example is \catcode‘\<=1, \catcode‘\>=2, which define the characters ‘<’ and ‘>’ as group delimiters. This does not change the definition of the braces, so now a group may be specified by {...>.

Here are some notes about catcodes:

- A character created by ‘^^’ is assigned a catcode in the mouth, but a character created by \char is not assigned a catcode at all! This is because \char is a primitive and is thus executed in the stomach. Executing \char always creates a character of text, which is typeset.

- Code 14 (comment) causes the rest of the input line to be ignored, is itself ignored, and no end-of-line character is appended to the line.

- A space following an active character is not ignored.

- Category 9 is always ignored, but can be used to delimit a control sequence name. Thus if we change the catcode of ‘x’ to 9 (by \catcode‘\x=9), future occurrences of ‘x’ will be ignored, and the string ‘NxO’ will be typeset as ‘NO’. However, a macro \abc can now be expanded by saying \abcx.... The ignored character ‘x’ serves to delimit the name \abc the way a space normally does. When TEX starts, the only ignored character is the ASCII ⟨null⟩ (^^@).

Exercise 4: What's the reason for category 9 (in other words, why type a character and then ignore it)?

Answer. When TEX is used to produce graphics, we sometimes want to suppress all spaces (see example on [390]). This can be done by \catcode‘\ =9. A carriage return can be ignored by saying

```
\catcode‘\
=9
```

on two separate lines. Also, certain control characters can sometimes be added to a file when it is transmitted between computers, and they should be ignored by TEX.

- Each blank line becomes a \par token, so there may be several consecutive such tokens, of which only the first is executed by the stomach.

Category 15 (invalid character) is initially assigned to the ASCII ⟨delete⟩ (^^?). TEX complains when it reads an invalid character.

— 16. Commands —

Many different commands are available in TEX, and they can be classified in two ways:

- A command can be classified as a primitive, a character, or user-defined. The latter category is further classified into a macro or an active character.
- A command can also be classified as either horizontal, vertical, or neither.

Beginners learn very quickly that a command should start with a ‘\’. However, an active character is also a command, and even a character of text (catcodes 11, 12) is one (see [267]). When a command is used very often, we want its name to be as short as possible, so we define it as an *active character* (catcode 13). It then becomes a one-character command, without even a ‘\’.

It is useful to consider a character of text (other than a space) a command. Such a command tells TEX to start a new paragraph or, if it is already in H mode, to typeset the character. A character is thus a *horizontal command.* If we want TEX to treat a character as not horizontal, we can place it in an \hbox. Interestingly, an \hbox is not inherently horizontal (but neither is it inherently vertical).

A command starting with a ‘\’ is called a *control sequence.* If it consists of letters only (catcode 11), it is called a *control word*; if it consists of a non-letter (any catcode $\neq$ 11), it is called a *control symbol.* Any spaces (or end of line) following a control word are ignored by TEX since it assumes that they are there only to delimit the word. However, a control symbol can only have one character following the ‘\’, so there is no need for any delimiters. Spaces following a control symbol are not ignored, and a situation such as ‘\?1’ is interpreted as the control symbol \? (normally undefined), followed by a ‘1’ (which is normally typeset).

The ignore-space rule above, however, applies only to characters coming from the input file; if a control sequence comes from a token list [39], a space following it will not be ignored.

Exercise 5: Devise tests to prove the preceding statements!

Answer.

```
\def\abc{‘} \abc ’ \abc\ ’
\def\?{\message{ok}} ‘\?1’ ‘\? ’
\toks0={\abc}
\toks0=\expandafter{\the\toks0 ’} \showthe\toks0
```

The concept of a delimiter is worth a little discussion. If we want to expand a macro \abc we can say ‘\abc␣’. However, \abc1... is also okay. The ‘\’ tells TEX that a control sequence starts, the ‘a’ tells it that this is a control word (just letters), and the ‘1’ delimits the string of letters. After TEX reads the ‘1’ it backspaces over it, executes \abc, and rereads the ‘1’. The point it that the ‘1’ is read twice and, when it is first read, it is not assigned a catcode.

This is a subtle point that may, sometimes, lead to errors. Consider the following:

```
\def\abc{\catcode‘\%=11 }
\abc%
\abc%
```

The first line defines \abc as a macro that makes the ‘%’ a letter. The second line uses ‘%’ to delimit the string of letters abc. The ‘%’ is thus read twice. When it is first read, it is not assigned a catcode, and \abc has not been expanded yet. When the ‘%’ is read again, \abc has already been expanded, so the ‘%’ is assigned a catcode of 11, which causes it to be typeset. When the third line is input, the ‘%’ already has catcode 11, so it is considered a letter. TEX thus ends up with the string ‘abc%’, and tries to expand macro \abc% which is normally undefined. The result is the message ! Undefined control sequence \abc%.

Exercise 6: Why is there a space following the ‘11’ in the definition of \abc above, and what happens without that space.

Answer. This is a result of the general rule that says that a number should normally be terminated by a space. TEX considers the space a terminator, so it does not get typeset. To better understand this rule, try the following:

```
\count20=20%
1stop
```

This example sets `\count20` to 201 and typesets 'stop'. The reason is that, after reading the '20', TEX reads ahead, hoping to find more digits. It finds the '1' since the '%' does not terminate a number.

Without the space, our three-line example is executed in the following steps:
1. The first line is read, and `\abc` is defined.
2. The second line is scanned. The '%' is read, which terminates the name `\abc`. The macro is expanded and the last thing, of course, is the '11'. Therefore, before executing the expansion, TEX reads the next character, hoping to find more digits. Since the next character is the '%', TEX skips to the next line and starts expanding `\abc`, still hoping to find more digits. The expansion of `\abc` on the third line, however, does not start with a digit.
3. At this point, TEX realizes that there are no digits following the '11' (from the second line). It therefore executes `\abc` from the second line, which changes the catcode of '%'.
4. Next, `\abc` from the third line is executed, which does not change a thing.
5. The '%' of the third line is reread and, since it is now considered a letter, it is typeset. (End of answer.)

There is an important difference between a character and a control sequence. A character has a catcode attached to it, which tells the gullet and the stomach what to do with the character. A control sequence has no catcode and may be redefined at any time, so the gullet has to look up the current definition before it can expand the control sequence.

A command may have arguments. Thus `\kern` must be followed by a dimension, and a '`^`' in math mode must be followed by the superscript. Sometimes the arguments are optional (the '`=`' in an assignment is a typical example), and sometimes there is a choice of arguments (`\leaders` should be followed [281] by either a ⟨box⟩ or a ⟨rule⟩, so it can be followed by one of the following: `\box15 \copy16 \vsplit17 \lastbox \hbox \vbox \vtop \hrule` or `\vrule`). The `\font` command [16] makes for an interesting example. It starts with `\font`\⟨name⟩=⟨file name⟩, followed by the optional arguments '`at` ⟨size⟩' or '`scaled` ⟨factor⟩'. The words '`at`', '`scaled`' are called *keywords*, and don't have a '\'. See [61] for a complete list of keywords.

In a command such as `\setbox0=\hbox{...}`, the arguments are '`0=\hbox{...}`'.

— 17. Assignments —

An assignment [275] is any command that assigns a new meaning to a control sequence or to an internal quantity. Examples are `\def`, `\hsize=...`, `\font\abc=...`, `\setbox0=...` & `\advance\x...`. Note that the '=' is always optional. Assignments are examples of commands that are executed in the same way regardless of the current mode.

Exercise 7: What other commands are executed in a mode-independent way?

Answer. See list on [279–281].

— 18. Lists —

This is another important concept. The contents of a box is a list, as is the contents of a math formula. A list is made up of items such as boxes, glue & penalties.

TEX is assembling a horizontal list when it is in H mode (building paragraphs) or in RH mode (building an `\hbox`). The items of such a list are strung horizontally, left to right, and must be H mode material [95]. In H mode, a list is terminated when TEX reads a `\par` (or a blank line), or anything that's vertical in nature (such as a `\vskip`). In RH mode, TEX terminates a list when it finds the '`}`' of the `\hbox`. If it finds inherently vertical material in this mode, it issues an error. Examples of horizontal commands are `\vrule`, `\valign`, `\char` and a character of text (see [283] for the complete list).

A character of text is a horizontal command whose meaning is: Add me to the current horizontal list or, if there is no current H list, start one with me as the first item.

TEX is assembling a vertical list when it is in V mode (between paragraphs) or in IV mode (building a `\vbox`). The items of such a list are stacked vertically, top to bottom, and must be V mode material [110]. In V mode, a list is terminated when TEX sees an inherently horizontal command, such as an `\hskip` or the first character of the next paragraph. In IV mode, TEX terminates a list when it finds the '`}`' of the `\vbox`. If it finds inherently horizontal material in this mode, it issues an error. Examples of vertical commands are `\hrule`, `\halign` & `\end` (see [286] for the complete list).

Kern is an interesting example of an item that may appear in either horizontal or vertical lists. Even though the same command, `\kern`, is used, it has different meanings in those lists. Kern is essentially rigid glue with the difference that TEX does not break a line or a page at a kern. Thus if two boxes are separated by a kern, they will remain tied. Of course, if the kern is followed by glue, a break is possible at the glue.

— 19. Whatsits —

A whatsit is an item that may appear in either a horizontal or a vertical list. It has no dimensions and signifies an operation that should be delayed. The paragraph builder and page builder scan lists submitted to them and execute certain whatsits. There are three types of whatsits:

▪ `\special`. This command requires two arguments. They are first stored in the MVL, and end up being written, at `\shipout` time, to the DVI file. They are interpreted and executed by the printer driver. Individual printer drivers support different `\special` arguments, so `\special` is an example of a non-compatible command. A document using it may only be printed with certain printer drivers. A typical example is `\special{postscript` ⟨postscript commands⟩`}`. When the printer driver finds this in the DVI file, it sends the ps commands to the printer, so that special printing effects can easily be achieved.

▪ The three non-immediate output commands `\openout`, `\closeout` & `\write`. They are also stored in the MVL and are executed later, at `\shipout` time. The reason for their delayed execution is that they may have to write the page number on an output file, and that number is only known in the output routine.

▪ The `\language` and `\setlanguage` commands [455] also produce whatsits each time the language is switched in the midst of a paragraph (e.g., from Icelandic to Serbo-Croatian). These whatsits are stored in memory with the rest of the current paragraph, while the paragraph text is being read in H mode. When the paragraph builder typesets the paragraph (determines the line breaks), the whatsits are used to select the set of hyphenation rules appropriate for each language.

— 20. Parameters —

Many numeric values are used by TEX, that a user may want to examine or modify. Those values are, therefore, given names, and are considered *parameters* of TEX (different from macro parameters). They are all listed on [272–275]. A typical example of a parameter is `\hsize`. Its value is a dimension, so it may be used whenever a dimension is necessary, as in `\hbox to\hsize` or `\hskip\hsize`. It may also be assigned a new value by `\hsize=3.5in` (however, the '=' is optional).

The rule is that the value of a parameter is used if its name appears in a context where such a value is needed. The value is changed if the name appears in any other context. A common error is a sentence such as 'The width of a line is normally `\hsize` but,...' When TEX sees `\hsize`, it is in the midst of typesetting our sentence, so it is in a context where it does not need a dimension. It assumes, therefore, that `\hsize` is an attepmt to modify `\hsize`, and reads ahead, expecting the new value. Finding the word 'but' instead, it complains of a missing number. The correct sentence should, of course, be 'The width of a line is normally `\the\hsize` but,....'

— 21. Macros —

This material is intended for users who rarely use macros but are otherwise experienced. A macro is a list of tokens that's been given a name, typically because it is used a lot in certain documents. For example, if the following is used many times in a document '`\medskip$\bullet$\enskip`', it can be given a name, such as `\section`, and defined as a macro by '`\def\section{\medskip$\bullet$\enskip}`'. After this definition, the macro can be *expanded* by simply saying '`\section`'. To expand a macro means to expand the tokens that constitute it. Those tokens are also called *the replacement text* of the macro, since they replace the macro name during expansion.

However, the feature that makes macros so useful and powerful is the use of parameters. By using parameters, each expansion of the same macro may be different. Our macro above may be extended by adding a parameter, such as the name of the section. After '`\def\section#1{\medskip$\bullet$\enskip{\it#1}}`', each expansion must supply some text that will be typeset as the name of the new section. Thus, e.g., '`\section{Advanced Techniques}`' will expand the tokens of the macro, and replace the parameter `#1` by the argument 'Advanced Techniques'.

Note the difference between *parameters*, which are formal and have no fixed value, and *actual arguments* which are text and commands that TEX can process. The notion of parameters also generalizes the concept of a token. Up until now, a token was either a control sequence or a single character. From now on, a token can also be a macro parameter, something of the form '`#x`', where `x` is one of the digits $1, 2, \ldots, 9$. A macro can have up to 9 parameters and, each time it is expanded, arguments must be supplied to replace each parameter.

Our first macro is now extended by adding another parameter, '`#2`', whose value is the section number. Defining '`\def\section#1#2{\medskip$\bullet$\enskip{\bf#2.\thinspace}{\it#1}}`', each expansion must have the form '`\section{Advanced Techniques}6`'. The second argument, '6', is not enclosed in braces because of the following rule: "The argument of a macro is a single token (the next token in the input stream, except that it cannot be any of the braces), unless it is delimited or enclosed in braces." The first argument 'Advanced Techniques' is long, so braces are used to indicate its boundaries. The second argument is a single character (a token), so no braces are necessary; but what is a delimited argument?

When a macro is defined, each of the parameters `#1`, `#2`, can be followed by any characters (except, of course, '`#`' and '`{`') which are its delimiter. If this is done, then each argument in any of the expansions must be followed by the same characters. Our `\section` macro is now extended to include delimiters '`\def\section#1;#2.{\medskip$\bullet$\enskip{\bf#2.\thinspace}{\it#1}}`' Each expansion of the new macro must look like '`\section Advanced Techniques;26.`' The first argument is everything up to, but not including, the first ';'. The second argument is everything following the ';' up to the next period. The delimiters themselves are skipped and do not become part of the actual arguments.

A simple example of delimiters is '`\def\test1#1#2. #3\end{...}`'. The first parameter has no delimiter, so it must be either a single token or braced. However, it must be *preceded* by a '`1`'. The second token is delimited by the two characters '`.␣`' (not just a period), and the third one, by the characters `\end`. Since the delimiters are skipped, the `\end` is not expanded (what would happen if it were?) If we now expand '`\test10123. 45\end`', the parameters will be '0', '123', '45'. If the expansion does not provide the right delimiters—such as in '`\test210123. 45\end`', '`\test10123.45\end`'—TEX issues the error message

```
! Use of \test doesn't match its definition.
```

When a parameter is delimited, the argument may contain braces, but they must be balanced. Thus in: `\def\x#1\end{\message{'#1'}} \x 1{2\end3}4\end5`, the argument is '`1{2\end 3}4`'.

Delimited parameters may get complex and confusing (see example on [203]), so the `\message` command may be used to find out the precise value assigned to each parameter
`\def\test1#1#2. #3\end{\message{arg1=#1; arg2=#2; arg3=#3}...}`. Another useful debugging tool is the `\tracingmacros` command, discussed among advanced macro features.

In addition to `\def`, macros can also be defined by `\let`, `\chardef`, and as active characters. It has already been mentioned that an active character is a control sequence whose name is limited to a single character, without even a '\', but whose replacement text can be of any length. The `\chardef` command [44] is, in a sense, the opposite of an active character. It defines a control sequence whose replacement text is limited to just one number (in the range 0–255), but whose name can be any valid cs name.

Thus `\chardef\abc=98` or `\chardef\abc='\b` define `\abc` as a macro whose replacement text is the character code of 'b'. It is equivalent to `\def\abc{\char98}`. Also `\chardef\active=13` defines `\active` as a macro whose value is the number 13.

The `\let` command is still different. Its general form is `\let` ⟨control sequence⟩=⟨token⟩. It defines the control sequence as being identical to the token. Thus defining

```
\def\a{X}
\let\g=\a \def\k{\a}
\g\k
```

produces 'XX'. If we now redefine `\a`, the meaning of `\k` will change (since it was defined by `\def`) but `\g` will not change. Thus `\def\a{*} \g\k` produces 'X*'. The difference between `\def` and `\let` is now clear. `\let\g=\a` creates `\g` as an independent macro that does not change if `\a` is modifed. In contrast, `\def\k{\a}` Creates `\k` as a macro pointing to `\a`, so it is always dependent on `\a`.

— 22. Formats —

When TEX starts, there are no user-defined macros, and the only commands available are the primitives. There are about 300 of them. The primitive `\def` can be used to define macros, which are then added to the repertoire of available commands. When `\def` is executed in the stomach, it loads the replacement text of the new macro into a special table in memory.

When a large document, such as a book, is developed, many macros may have to be defined. Each time the document is typeset, all the definitions have to be prepared by the stomach and loaded in memory, which may be time consuming. In such a case, it is better to convert the macros into a *format*. A format is a collection of commands (macros and primitives), written in a special way on a file, to facilitate rapid loading.

A complete TEX system includes a program called `INITEX` [39] that is used to install TEX. `INITEX` is like TEX but can also prepare formats and hyphenation tables, and perform other tasks. To prepare a format, `INITEX` should be run, the relevant macros should be defined, and the `\dump` command [283] executed. This command dumps the table containing the macros from memory onto a file. That file is called a format file, and it can later be loaded fast simply be copying its contents directly into memory. A format file is loaded, like any other file, by the `\input` command.

A very useful format, the `plain` format, comes with every TEX implementation. It is written on file `plain.fmt` and it consists of about 600 macros that are described throughout the TEXbook, and are useful for general purpose typesetting. The `plain` format is also listed in [App. B]. Many commonly used commands, such as `\bye` % `\smallskip`, are part of this format.

When macros are developed for a specific document, they are normally used together with the `plain` format. It is easy to create a new format containing all the `plain` macros plus any user-defined ones. See [344] for directions.

TEXtures, a Macintosh TEX implementation, makes it particularly easy to create formats since it can execute the `\dump` command without any need for `INITEX`. It is also easy to load any existing format in TEXtures at the flick of the mouse, without having to execute any commands.

— 23. References —

1. Knuth. D. E, *Computers and Typesetting*, vol. E, Addison-Wesley, 1986.

2. Henderson, D, *Outline fonts with* METAFONT, TUGboat **10**(1) April 1989, 36.

3. Wujastyk, D, *The many faces of TEX*, TUGboat **9**(2) August 1988, 131.

4. Tobin, G. K. M, *The OCLC roman family of fonts*, TUGboat **5**(1) May 1984, 36.

5. Billawala, N, *Metamarks: Preliminary Studies for a Pandora's Box of Shapes*, Report STAN-CS-89-1256, Computer Science Department, Stanford University, 1989.

6. Siegel, D. R, *The Euler Project at Stanford*, Department of Computer Science, Stanford University, 1985

7. Haralambous, Y, *Typesetting Old German*, TUGboat **12**(1) March 1991, 129.

8. Fuchs, D, *The Format of TEX's* DVI *Files, Version 1*, TUGboat **2**(2), July 1981, 12.

> Do not ye yet understand,
> that whatsoever entereth in at the mouth
> goeth into the belly,
> and is cast out into the draught?
>
> — *Mathew 15:17*

> Now there are times when a whole generation is caught . . . between two ages, between two modes of life and thus loses the feeling for itself, for the self-evident, for all morals, for being safe and innocent.
>
> — *Hermann Hesse, Steppenwolf*

The Components of TEX

Joachim Schrod
Detig · Schrod TEXsys

March 1991

Abstract

TEX needs a great amount of supplementary components (files and programs) of which the meaning and interaction often is unknown. This paper explains the components of the kernel system TEX that are visible for the TEX user and their relations.

1 About this Report

TEX is a typesetting system which offers authors easy usage of powerful typesetting features to produce printed matter which is the state of the art of computer typesetting. This is, however, not done by the TEX program alone: A significant number of supplementary programs and files together form the complete typesetting and authoring system. Along with the programs that belong to TEX directly, there exist two other major programs which were built by DONALD KNUTH in connection with TEX and must be included in an explanation of the full system: METAFONT, for the generation of fonts, and `WEB`, a documentation and developing 'language' for programming. TEX and METAFONT are written in `WEB`.

This text describes this 'kernel' TEX from a user's viewpoint: at the end you should have an overview of the ingredients of the TEX system, and about the files and support programs that are essential for you as a user. This will not, however, be an introduction to the capabilities of TEX or how you may run TEX on your computer.

I will use marginal notes to identify the places where terms are explained for the first time. Abbrevations for file types – usually identified by common suffixes or extensions – are set in a monospace type ("`typewriter`") and those abbrevations are put into the margin, too. Please note that these abbreviations are sometimes not identical with the file extensions (see also Table 1).

This report is the start of a series that describes the subsystems mentioned above and their respective components. In that series each report will focus on one subsystem in a special point of view; it should not result in gigantic descriptions which tell everything (and then nothing). In my opinion the following reports will be of interest:

- the structure of a standard installation of TEX
- `DVI` drivers and fonts
- possibilities of graphics inclusion in TEX documents
- the components of METAFONT
- the structure of a standard installation of METAFONT
- `WEB` systems — the concept of Literate Programming
- other (though not yet planned) themes of interest are perhaps
  - differences between TEX and DTP systems
  - the way how TEX works (there exists some good books on this topic!)
  - the limits of TEX
  - TEX as a programming language

The reports will be published in this sequence.

2 What is TEX?

TEX is a typesetting system with great power for the typesetting of formulae. Its basic principle is that structures in the document are marked and transformed into typeset output. Providing such information about the structure of a document is known as *markup*. If the marks describe the look of the document, it is called *optical markup*, while, if document structures are marked, it is called *logical markup*. TEX provides both forms of

Previous revisions of this paper appeared in "Die TEXnische Komödie" and in "Baskerville."

Actual address: Joachim Schrod, Detig · Schrod TEXsys, Kranichweg 1, D-6074 Rödermark, FR Germany, Email: `xitijsch@ddathd21.bitnet` markup, i. e., exact control of the layout of parts of the document and their positioning as well as the markup of the structure of formulae or document components. The logical markup is mapped to the optical one by TEX so that layout may serve for the identification of structures by the reader.[1]

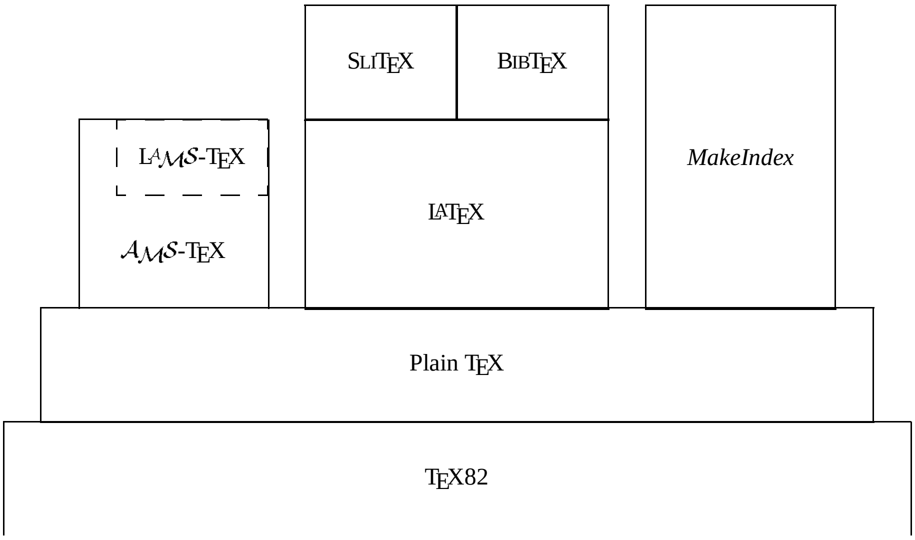
The kernel of the TEX typesetting system is the formatting program TEX82, which is often simply called TEX. This usage shall be adopted here whenever the difference between the complete system and the formatter is unimportant or obvious. TEX82 is a big monolithic program which is published in the book *TEX: The Program* by DONALD KNUTH. Its features may be separated into two levels:

1. TEX82 formats text, i. e., it breaks it into paragraphs (including automatic hyphenation) and produces page breaks.
2. It provides the programming language TEX which incorporates a macro mechanism. This allows new commands to be built to support markup at a higher level. DONALD KNUTH presents an example in the TEXbook: Plain TEX. A collection of macros which supports a special task and has (hopefully) a common philosophy of usage is called a *macro package*.

High level features for optical markup, as represented by Plain TEX, allow one to build additional levels leading to full logical markup. At the moment, two macro packages for logical markup are widespread: AMS-TEX and LATEX. Both systems are built on top of Plain TEX to greater or lesser extents and the user can use the optical markup of Plain TEX in addition to logical markup if desired. This results in the effect that the author can use a mixture of structural information and explicit layout information – a situation with a high potency of features that nevertheless can (and does) lead to a lot of typographic nonsense!

As TEX82 was built only for typesetting texts and to allow the realization of new markup structures, many features are lacking which are required by authors. To provide features like the production of an index or a bibliography or the inclusion of graphics, additional programs have been written, which use information from a TEX82 formatting run, process them, and provide them for the next TEX82 run. Two supplementary programs are in widespread use and available for many computer/operating system combinations: BIBTEX, for the production of a bibliography from a reference collection, and *MakeIndex*, for the production of an index.

A special case of the processing of information provided by a TEX run is the production of a table of contents or the usage of cross references in a text. For this only informations about page numbers, section numbers, etc., are needed. These are provided by TEX82 and can be processed by TEX82 itself, so TEX82 is used as its own post processor in this situation.



Figuur 1: *The Components of TEX*

[1] Layout – and book design in general – do not represent useless beauty. A good book design must first support the understanding of the content to produce readable text. So it is *æsthetic* in its best sense, since it connects form and contents and builts a new quality.

We have now seen that the TEX typesetting system is a collection of tools that consists of the typesetting 'engine' TEX82, macro packages (maybe several that are based on others) and supplementary programs, used together with these macro packages. This relation is illustrated by Figure 1.

3 Formatting

The formatting process of TEX needs information about the dimensions of characters used for the paragraph breaking. A set of characters is grouped in *fonts*. (But this is a simplification as the notion "font" should be used for the realization of a type in a fixed size for a specific output device.) The dimensions of the characters of a font are called *font metrics*.

The format in which the font metrics are used by TEX was defined by DONALD KNUTH and is called `TFM` format ("*TEX font metrics*"). In this format, every character is descibed as a *box* with a height, a depth, and a width. TEX only needs these measurements, it is not interested in the shape of the character. It is even possible that the character may extend outside the box, which may result in an overlap with other characters. The character measures are specified in a device independent dimension because TEX processes its breaking algorithm independent of any output device.

During paragraph breaking, TEX hyphenates automatically, which can be done in an almost language-independent way. For the adaption to different languages, *hyphenation patterns* are needed to parametrize the hyphenation algorithm.

The result of a TEX formatting run is a `DVI` document, in which the type and position on the page are specified for each character to be output. The resolution that is used is so small that every possible output device will have a coarser raster, so that the positioning is effectively device independent. The `DVI` document specifies only types, not the fonts themselves, so that the name `DVI`[2] ("*device independent*") is accurate. To make the result of the formatting run available, the `DVI` file must be output by a so-called `DVI` driver on the desired output device.

If problems occur during the formatting, error messages or warnings are output on the terminal. *Every* message that appears on the terminal will also be written into a protocol file named `LOG` file. In this `LOG` file additional information may be placed that would have been too verbose for the output to the terminal. If this is the case, TEX will tell the user so at the end of the formatting run. The messages of TEX are not built in the program, they are stored in a (string) `POOL` file. These messages must be read in at the beginning of a run.

4 Macro Packages

The basic macro package is Plain TEX, developed by DONALD KNUTH together with TEX82. It parametrizes the TEX82 typesetting machine so that it can typeset English texts with the Computer Modern type family. Additionally, Plain TEX provides optical markup features. Plain TEX is available as one source file, `plain.tex`.

All other macro packages known to the author are based on Plain TEX, i. e., they contain the source file `plain.tex` either originally or with modifications of less important parts. Next to Plain TEX, the most important (free) available macro packages are AMS-TEX by MICHAEL SPIVAK and LATEX by LESLIE LAMPORT. Other free macro packages are often of only local importance (e. g. BlueTEX, TEXT1, or TEXsis) or are used in very special environments only (e. g. `texinfo` in the GNU project or `webmac` for `WEB`). Important commercial macro packages are MacroTEX by AMY HENDRICKSON and LAMS-TEX, also written by MICHAEL SPIVAK.

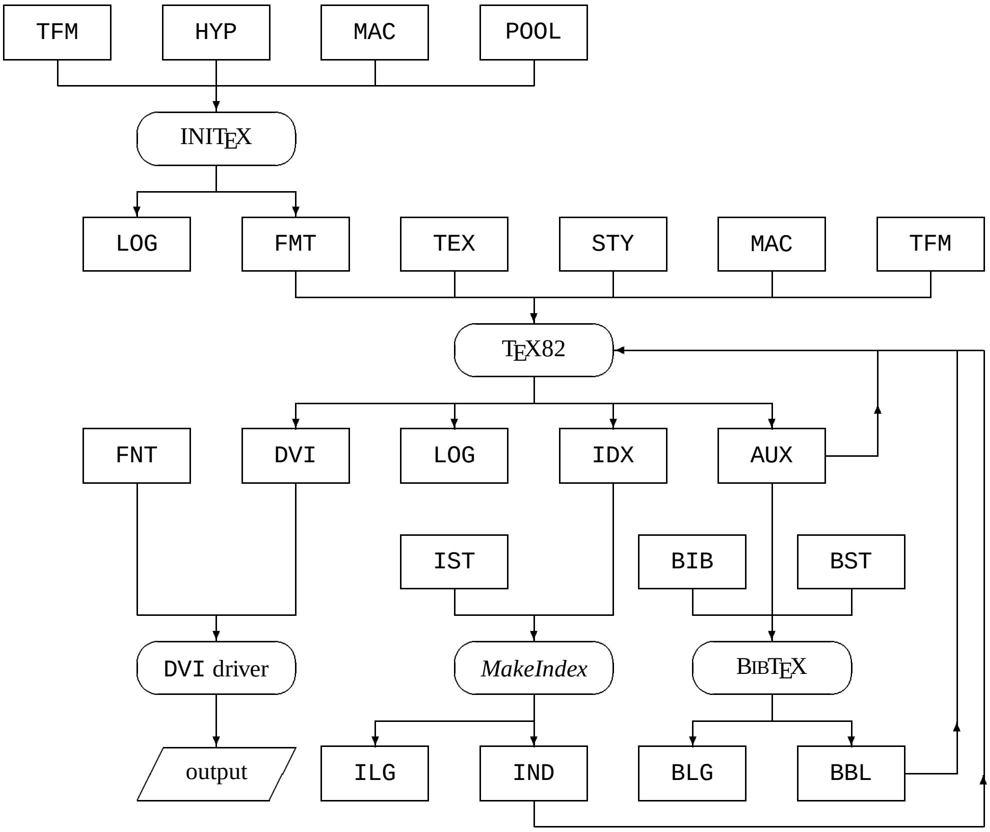
These macro packages usually consist of a kernel that provides additional markup primitives. With such primitives, *document styles* can be built which realize logical markups by a corresponding layout. This layout can often be varied by *sub-styles* or *style options* which may also provide additional markups.

The macro packages produce *supplementary files* which contain information about the page breaks or the document markup. This information may be used by support programs – e. g., the specification of a reference from a bibliography database or the specification of an index entry with corresponding page number for the construction of an index. A special case is the information about cross references and headings for the building of a table of contents, as this information can be gathered and reused by TEX directly.

SLITEX is a special component of LATEX for the preparation of slides with overlays. In TUGBoat volume 10, no. 3 (1989) LAMS-TEX was announced, which will provide the functionality of LATEX within AMS-TEX. MacroTEX is a toolbox of macro "modules" which may be used to realize new markups but, as it became available only short time ago, it is not yet widespread.

For the usage of these (and other) macro packages, one must check whether they need additional fonts which do not belong to the Computer Modern type family. For LATEX, e. g., fonts with additional symbols and with invisible characters (for the slide overlays) are needed, while AMS-TEX needs several additional font sets with mathematical and Cyrillic characters.

[2] This name is a problem because "DVI" is a trademark of Intel Corp. now, but the name `DVI` for TEX output files pre-dates this.



Figuur 2: *The Connection of Components and File Types*

5 Support Programs

Only two support programs will be discussed here: BIBTEX by OREN PATASHNIK for the preparation of bibliographies and *MakeIndex* by PEHONG CHEN and MICHAEL HARRISON for the preparation of a sorted index. For both tasks exist other, funcionally equivalent, support programs. But the abovementioned are available on many operating systems, and have an “official” state as they are LESLIE LAMPORT encourages their usage with LATEX in his documentation, and the TUG supports them for general use.

There is no totally portable mechanism for the inclusion of general graphics in TEX documents, so that there are no machine independent support programs available.

BIBTEX is used to handle references collected in BIB files. TEX produces supplementary files which contain information about the required references, and BIBTEX generates from them a sorted bibliography in a BBL file which may be subsequently used by TEX. The kind of sorting and the type of cite keys are defined by *bibliography styles*, specified in BST files. The messages of a BIBTEX run are written to a BLG logfile.

MakeIndex reads an IDX support file that contains the index entries and the according page numbers, sorts these items, unifies them and writes them as TEX input in an IND file. The formatting style may be specified by an *index style*. The messages of a *MakeIndex* run are written to a ILG file.

6 Performance Improvements

Much of the work that TEX82 has to do is the same for every document:

1. All text has to be broken into lines. Text pieces in the same language are hyphenated with the same hyphenation patterns.
2. The basic markups of the corresponding macro packages must be available.
3. The required font metrics are much alike for many documents, as the font set used usually doesn’t differ that much.

To improve TEX’s performance, hyphenation, markup, and font metrics descriptions are converted from an external, for (1) and (2) textual, representation into an internal representation which can easily be used by TEX82. It is sensible to do this transformation only once, not for every document. The internal representation is stored in a FMT file. The storing is done with the TEX command `\dump`, so that FMT files often are

FILE TYPE	EXPLANATION	FILE IDENTIFICATION (SUFFIX, EXTENSION, ETC.)
`TEX`	Text input	`tex`, `ltx`
`DVI`	TEX82 output, formatted text	`dvi`
`LOG`	TEX82 log file	`log`, `lis`, `list`
`HYP`	Hyphenation patterns	`tex`
`TFM`	Font metrics	`tfm`
`POOL`	String pool	`pool`, `poo`, `pol`
`FMT`	Format file	`fmt`
`MAC`	TEX macro file	`tex`, `doc`
`STY`	TEX style file	`sty`, `tex`, `st`, `doc`
`AUX`	Support files	`aux`, `toc`, `lot`, `lof`, `glo`, `tmp`, `tex`
`BIB`	Reference collections	`bib`
`BBL`	References or bibliographies	`bbl`
`BLG`	BIBTEX log file	`blg`
`BST`	BIBTEX style file	`bst`
`IDX`	Unsorted index	`idx`
`IND`	Sorted index	`ind`
`IST`	Index markup specification	
`ILG`	*MakeIndex* log file	`ilg`

Tabel 1: *File Types*

called "*dumped formats*." A `FMT` file can be read at the beginning of a TEX82 run and is thus available for the processing of the actual text.

As the creation of a `FMT` file is done infrequently – usually for the update of a macro package – the formatting of texts can be done with a reduced version of the TEX82 program that doesn't contain the storage and the program parts for the transformation of the hyphenation patterns and for the dumping. The complete version of TEX82 is needed in an initialization phase only and therefore called INITEX. Additional improvements of the performance can be reached by the usage of production versions of TEX82 from which parts for statistical analysis and for debugging are stripped.

TEX versions that have no dumped formats preloaded, have the ability to load a dumped format (i. e. a `FMT` file), and have no ability to dump a `FMT` file (i. e., they are not INITEX) are often called VirTEX, which stands for *virgin TEX*.

7 Connections Between File Types and Components

In the above sections, the components of the TEX authoring system were described, and the files that are read or written by these components mentioned. The connections between them all is demonstrated graphically in Figure 2. In this graphic, file types are represented by rectangles, and programs by ovals. The arrows mean "is read by" or "is produced by." The abbreviations of the file types are explained in Table 1, which also lists the file identifications (suffixes or extensions) that these files usually have (but note that other file identifications are also in use).

Acknowledgements

I would like to thank CHRISTINE DETIG who was so kind to provide the English translation. NELSON BEEBE suggested performing the translation. KLAUS GUNTERMANN made valuable comments on the first (German) version. NICO POPPELIER contributed a new version of figure 2, better than my original one.

LaTeX 3 project

Frank Mittelbach

Stockholm
18 November 1991

LaTeX 2.09 ↪ LaTeX3

Frank Mittelbach
Electronic Data Systems
Federal Republic of Germany

Asked to speak about LaTeX2.09 → LaTeX3 I will try to give a you a picture of the history, the current state, and the future of the LaTeX3 project.

1. Historical Remarks

Whenever the future is somewhat unpredictable it seems wise to take a look into history—to find out what is already achieved and what remains to be tackled.

From the history of the LaTeX3 project we will first take a look at the growing bulk of syntax descriptions and (partial) implementations that are the results of three years work.

Milestones:
Syntax and implementation

1988
- Some bug fixes send to Dr. Lamport
- Four page sketch of NFSS

1989
- First implementation of NFSS
- Implementation of `amstex.sty`

1990
- New tabular implementation by D. Duchier
- First attribute prototype (thrown away)
- First kernel prototype
- First recovery/help prototype

1991
- Second kernel prototype
- Sketches for style designer interface
- Second description of the attribute concept
- Extended description of the help facility
- Syntax for extended NFSS
- Third kernel prototype
- Release of LaTeX 2.09 international with NFSS support

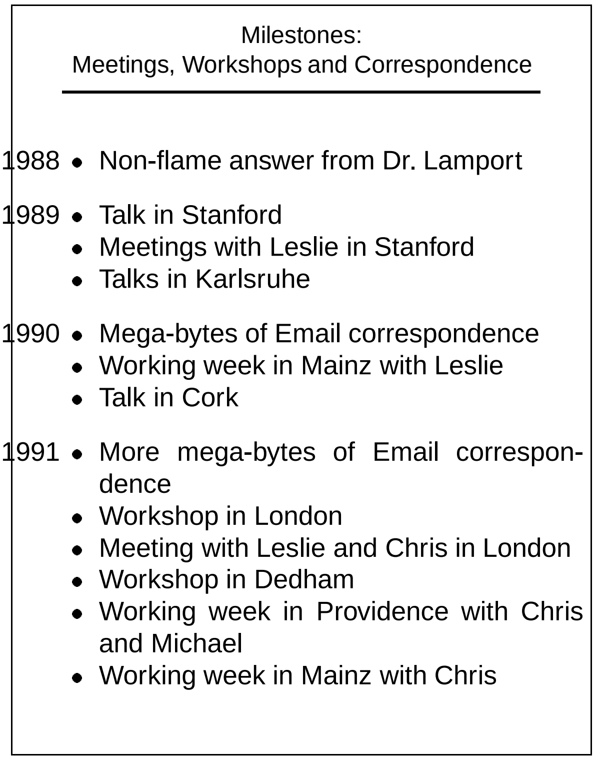
The LaTeX3 project was initiated at the Stanford annual meeting in 1989. But first vague plans were already formulated in 1988 when Rainer Schöpf and I, after sending several pages of bug fixes for LaTeX 2.09 to Leslie Lamport, received a positive answer.

Given the original goals for a reimplementation described in the Stanford paper, nearly everything seems to be achieved.

- With NFSS there is a far more general font selection available. The extended syntax also provides for font scaling (prototype implementation done).
- With `amstex.sty` the mathematical capabilities of LaTeX have reached the standard of AMSTeX.
- With the new tabular implementation by Denys Duchier (that superseeds `array.sty`) and valuable suggestions by several others tabular processing has reached very high quality.
- With the new help/recovery concepts a safe and easy to learn environment is available for novice users.

- With the partially finished concept for specifying attributes to environments and functions a more flexible input language is available. This also allows easy conversion from SGML to LATEX3 DTDs.

How did this happen?



Milestones:
Meetings, Workshops and Correspondence

1988
- Non-flame answer from Dr. Lamport

1989
- Talk in Stanford
- Meetings with Leslie in Stanford
- Talks in Karlsruhe

1990
- Mega-bytes of Email correspondence
- Working week in Mainz with Leslie
- Talk in Cork

1991
- More mega-bytes of Email correspondence
- Workshop in London
- Meeting with Leslie and Chris in London
- Workshop in Dedham
- Working week in Providence with Chris and Michael
- Working week in Mainz with Chris

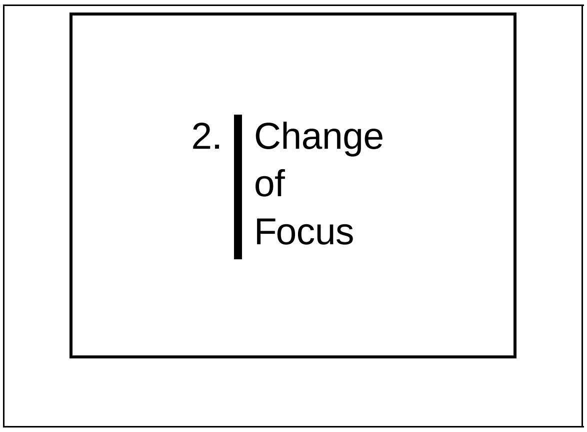
All this work has been carried out in the free time of several individuals and involves, as you can see, some enthusiasm to keep the project alive. So far, more than thirty people have contributed in one way or the other.

One of the major problems is to bring people together to discuss the open questions and find new solutions. This must also involve people outside the project since we do need the opinion and experience of typesetters, publishers, etc. to eliminate the flaws in the system and find new and better solutions.

In this regard both the London and the Dedham workshop have been a great success but further workshops of this kind are definitely necessary to provide LATEX3 with a suitable designer interface.

So why is this project in our opinion still being at its start? Because we have learned that our original goals haven't touched the real problems as we see them today.

This has lead to a

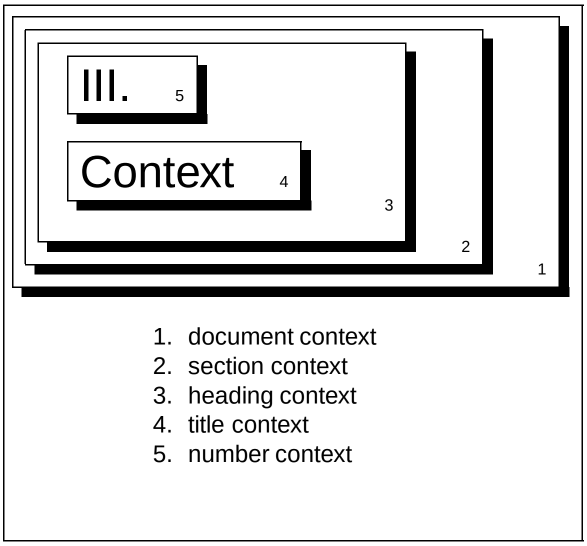


2. Change of Focus

We now feel that one doesn't gain much by providing more and more specialized style files that solve this or that special problem. Instead, we think that the major effort in the future has to go into the design of a suitable style interface that allow easy implementation of various layouts. (Easy, of course, is relative: easy compared to the complexity of the task.)

This change of focus implies
- The development of a new internal language that is more suited to express visual components of the layout process.
- The development of high-level generic functions that allow express most commonly used layout components in an easy way.
- The development of a model for specifying and modifying parameters that influence the layout.

Since the syntax for this internal language is still changing on a daily basis and generic functions mostly depend on it, I like to concentrate today on the model for parameter setting.



1. document context
2. section context
3. heading context
4. title context
5. number context

The slide already shows our main idea for maintaining parameters in the LATEX3 system.

On every point in the document we are in some context that is given by the the nesting and sequencing of entities processed so far.

The major idea of the new system is to allow the specification of parameters within such contexts in a very general way. For example, it is possible to redefine the behavior of lists within footnotes by specifying the values of list-parameters in the context of “footnotes” differently from those applied in the context of, say “floats”.

Parameter is meant in a very general way, e.g., the code that some entity runs is internally a parameter, so that via this concept different generic functions can be run in different contexts.

The concept of a context
Some observations

- The nesting of entities forms the major component for describing layout via contexts.
- The specification of layout by a sequence of contexts is important.
- The context of some entity in a document is not simply given by nesting and sequencing of surrounding entities.
- The context of some entity has a logical and a visual component. The visual component depends on the formatting of other entities.

As a further example, the layout attributes for a table entity in a float may be different then for tables in the main text.

Sequencing is, for example, important in heading→heading situations where intermediate spacing and penalties change if headings follow directly after each other, in list→text/par situations and many other places.

Take, for example, the situation of some footnote or float that appears in a list and itself contains a list. Because of an improper handling of contexts in the current LATEX the inner list is typeset as a second level list. In other words an entity must be able to (partially) forget about its context, or more generally must be able to manipulate its context.

The fourth point is of theoretical nature. All of the currently available formatters format document entities in a predetermined visual context, i.e., they assume that the visual context can be determined by the logical nesting and sequencing of entities. To a certain extend TEX is an exception as it applies dynamic programming to the process of paragraph formatting which involve recomputation of contexts for ligatures etc.

As an example for the wrong visual context consider a hyphen at the end of a page, that is avoided by TEX by moving the line instead of recomputing the paragraph.

The concept of a context
Some problems

- The user input is not at normalized document—it may contain hidden entities inside of user-defined shorthands that can not be prescanned easily.
- The specification of contexts by sequencing is important but partially restricted by the underlying TEX engine.
- Taking the visual component of contexts into account requires the use of a multi-pass system.

The problem here is that we don’t deal with normalized documents (where every entity is fully tagged) and therefore can not scan for further begin or end tags before we start typesetting. This means that certain decisions have to be taken without knowing what follows. The only solutions to this problem are

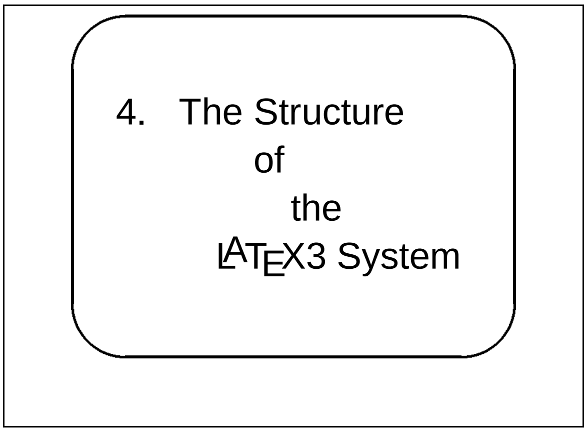
- dissallow the use of user defined shorthands
- the use of a two-pass system that normalize the document in the first pass
- the use of a multi-pass system that use sequencing information from the last run.

None of the solutions seem to be feasible for a system that uses TEX as the input language but should be explored further.

Deferring of the typesetting process is generally possible in TEX’s vertical mode where we can wait for the next `\everypar` to regain control.

But TEX has no build in mechanism to detect whether plain character material after a given point is about to be contributed to some horizontal list. Only after material has been contributed to the horizontal list one can deduce this fact by “dirty tricks” with special kerns. But this can be used only for interrupting a context sequence—the contributed material can not be manipulated further.

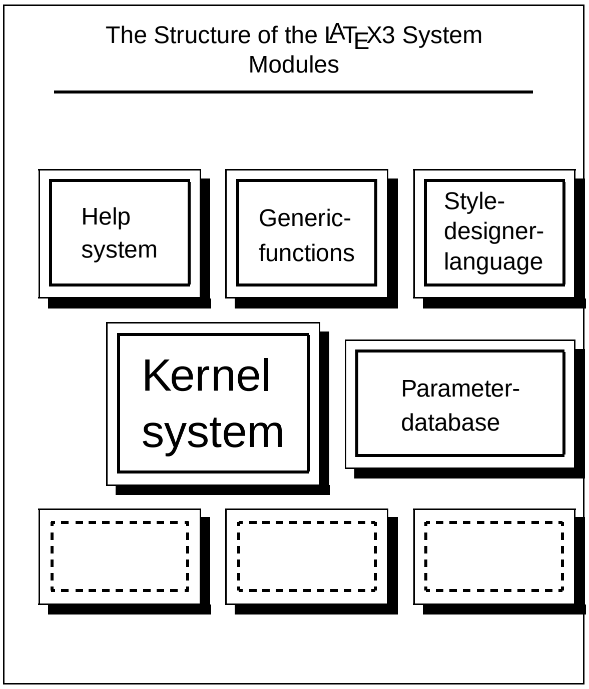
This will draw the boundary (beside processing time) between the ideal model and the real world. So let us now turn to the question of how this context-model is placed within the LATEX3 system.



4. The Structure of the LATEX3 System

There are many interwoven structures of the LATEX3 system that are worth talking about. In the following I will show how the already achieved results and the ideas about contexts fit together in an extensible modular system.

The LATEX3 system will consist of a kernel system that provides the basic data structures such as lists, stacks, etc. to program higher modules. It will also contain arithmetic functions for integers and dimensions, e.g., it will be possible to express relationships between individual parameters by specifying assignments that contain expressions.



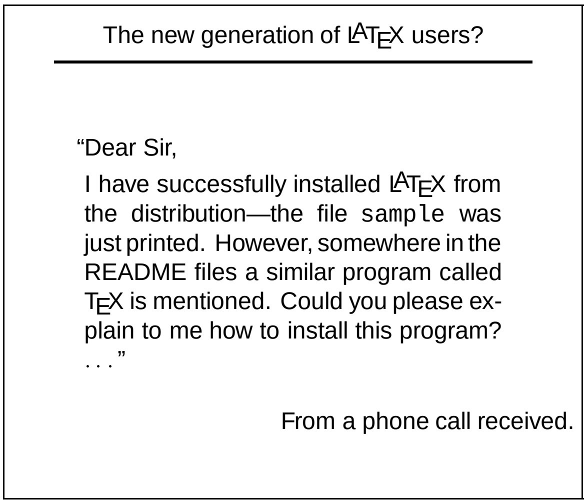
The Structure of the LATEX3 System Modules

On top of it manipulation functions for the parameter database and generic functions are build. They form the platform for the style designer language.

One important component of the system will be an interactive help system that allows extensive help texts as well as the possibility to define system reaction depending on user action. Help messages and such additional error-correcting code will be held in external files that are read in when an error is detected by the system. In this way an elaborate help and error correcting mechanism will be available while keeping the LATEX3 kernel compact.

We distinguish between document styles that are written in the style designer language (and will probably contain nearly no TEX code in the traditional sense) and additional modules that provide entities for specialized documents. This will include, for example, higher math and we hope that we can provide with the new kernel a programming interface that makes the development of further modules a possible task.

Let me close with this quotation from some unknown novice of LATEX.



The new generation of LATEX users?

"Dear Sir,

I have successfully installed LATEX from the distribution—the file `sample` was just printed. However, somewhere in the README files a similar program called TEX is mentioned. Could you please explain to me how to install this program? ..."

From a phone call received.

This might sound funny at first, but this extreme is not far from reality.

- Today, there are not many users who have a well-known understanding of the underlying system structure.
- Today, the majority of users use LATEX only. They usually have no knowledge of the TEXbook. This class of users can be nicely classified as "has heard of 'macros', but has never seen one".

TEX and LATEX as its major front-end has to compete with the so called Desktop-Publishing systems. To keep them alive we have to bridge the gap between the "implementor/wizard" type of user of the '80s and the new type that uses the system just as one tool out of manys without understanding its internals.

With the LATEX3 project we hope to achieve this goal as far as the front end is concerned. Our current eMails addresses are:

Frank Mittelbach:
`Mittelbach@mzdmza.zdv.Uni-Mainz.de`
Chris Rowley: `CA_Rowley@vax.acs.open.ac.uk`
Rainer Schöpf: `Schoepf@sc.ZIB-Berlin.de`

LaTeX Editing Support

Nelson H. F. Beebe

Center for Scientific Computing
Department of Mathematics
South Physics Building
University of Utah
Salt Lake City, UT 84112
USA
Tel: (801) 581-5254
FAX: (801) 581-4148

`beebe@math.utah.edu`

Contents

List of Figures

1 Introduction

The structured markup of LaTeX can be easy to read and understand, but tedious to type. Its syntax of environment groups bears a strong resemblance to the **begin**/ **end** groups of the Algol family of computer programming languages, which can be described by rigorous grammars that in turn permit the automatic construction of lexical analyzers and parsers, and structured editors to support programming in those languages [4, pp. 97–116, pp. 128–140, pp. 232-239]. This article describes a powerful facility for the preparation of LaTeX documents using the Emacs text editor.

The convention here is that material displayed in a `typewriter font` is computer output, a program name, or all or part of a file name. A sans serif font is used for key names and Emacs functions, and a *slanted font* for Emacs variable names.

Emacs is very likely the world's most powerful text editor, because it is highly programmable and customizable. The popularity of the original Emacs on the DEC PDP-10 architecture in the 1970s led to several reimplementations on other operating systems, particularly on UNIX and VAX VMS. The wide availability of the excellent implementation from the Free Software Foundation, known as GNU Emacs, has made Emacs the editor of choice for many programmers on machines more powerful than personal computers. Smaller Emacs-like editors have also been developed, including

- Jonathan Payne's `jove` (Jonathan's Own Version of Emacs) for UNIX, VAX VMS, IBM PC DOS, and the Apple Macintosh;
- `microemacs` and `freemacs` on IBM PC DOS;

- the commercial editors `epsilon` from Lugaru Software, Ltd. (5843 Forbes Avenue, Pittsburgh, PA 15217, USA), and `brief` from Solution Systems (541 Main St, Suite 410, South Weymouth, MA 02190, USA).

Several years ago, I developed substantial editing support in the TOPS-20 Emacs text editor for the Fortran and SFTRAN3 languages, and also for LaTeX. TOPS-20 Emacs was programmed in a very terse language called TECO, in which all 128 ASCII characters are assigned editing functions, and multi-character names provide thousands more.

As we acquired VAX VMS and UNIX systems running GNU Emacs, I reimplemented that editing support in GNU Emacs Lisp, which although much more verbose than TECO, is much easier to read and write.

The purpose of this article is to describe the capabilities of the LaTeX editing support. Before we get into that, however, some background about Emacs must first be presented. If you are already familiar with Emacs, you might want to skip ahead directly to the section **LaTeX support** beginning on page 94.

2 Emacs history

Emacs was developed by Richard M. Stallman starting about 1974 on a PDP-10 running MIT's ITS (Incompatible Timesharing System) operating system. The kernel of the editor was written in PDP-10 assembly language, which could be made to produce versions for DEC's TOPS-10 and TOPS-20 systems and Stanford's SAIL system as well, thanks to the availability of conditional compilation facilities. SAIL was the home of the original TeX development.

Because PDP-10 machines formed the backbone of the original Arpanet, the precursor to today's Internet linking millions of computers, and because Emacs' author believed fervently in the value of sharing of software, Emacs was soon in wide use throughout the Arpanet.

Like Topsy in *Uncle Tom's Cabin* [15, p. 277], PDP-10 Emacs just grew; the final version from the mid 1980s had about 150 libraries and perhaps 2000 editing functions. What made this possible was the architectural division of the editor into two parts:

- a modest kernel of low-level functions, like character insertion, deletion, and movement, string searching, and operating-system interface that have to be efficiently implemented for performance reasons, and
- a much larger collection of libraries written in a higher-level language, TECO (Text Editor and COrrector), that can be written, compiled, and debugged inside Emacs itself, thanks to the TECO interpreter and compiler included in the kernel. Libraries can be loaded on demand, or automatically when certain specified conditions are met.

Emacs' TECO was greatly extended beyond Dan Murphy's original TECO developed on early DEC PDP machines at MIT in the 1960s. TECO remains available today on DEC PDP-11 and VAX VMS systems, though it has largely fallen into disfavor because of its terseness, and because DEC's versions provide only limited support for screen-based editing.

The GNU Emacs reimplementation has a similar division of architecture, but its kernel is written in the C language for *portability*, and high-level functions are written in Lisp for *readability*. The Emacs Lisp interpreter is provided by the C-language kernel, and the compiler is written in Lisp. The Lisp dialect is peculiar to GNU Emacs, but is closely related to MIT's MACLisp. In particular, both use dynamic scoping, rather than the lexical scoping of Common Lisp; if you don't know why that is significant, read Stallman's article in [4].

Emacs supports simultaneous editing of multiple files in multiple buffers, with the screen divided horizontally and/or vertically to provide windows into one or more buffers. It also supports an extensive on-line help system, with self-documenting commands, and the full text of many user manuals accessible inside the editor.

All of this power requires a lot of code to support it, and about a megabyte of central memory to run satisfactorily; that is why GNU Emacs has not been ported to personal computer environments. At version 18.55, GNU Emacs consists of about 71400 lines of C code in 177 source files, 55300 lines of Lisp in 144 libraries, 584 functions written in C, and 2643 functions written in Lisp. The LaTeX support code described here consists of about 3900 lines of Lisp defining 82 functions and 48 variables, together with an additional 900 lines and 30 functions of auxiliary support.

Some people argue that such a powerful editor 'wastes machine resources'; I disagree—computers should work for people, not the reverse.

3 Emacs editing model

Some vendor-provided text editors, such as `edt` in VAX VMS and `vi` in UNIX, have two modes: command mode, and text insertion mode. The meaning of a key depends on which mode you are in, and some users find themselves frequently making mistakes because they type in the wrong mode.

Emacs does not make a distinction between these two. Instead, each character you type runs a function. The function attached to most printing characters just inserts them in the buffer, so entry of normal text works just like in other editors. However, certain keys invoke editing functions that do other tasks. For example, pressing the DELete key results in a call to a function that deletes the character on the screen immediately before the cursor.

The execution of editing functions in response to keystrokes requires character-at-a-time input. Some older mainframe operating systems do not support this, but all newer ones do.

Because most printing characters are needed for themselves, and most ASCII control characters have no useful printable representation, commonly-needed editing functions are assigned to control characters, with mnemonic significance where possible. For example, the character generated by holding down the control key while pressing the e key, represented by the compact notation C-e, runs a function that moves the cursor to the *end* of the current line, and C-d runs a function that *deletes* the character under the cursor.

There are only 33 control characters in ASCII, and more editing functions are needed. Some terminal keyboards in the 1970s provided an additional key, called a meta key, which worked like the control key, except that it added the 8th bit to the character value, generating a character in the range 128 . . . 255, thereby providing for another 128 keys to bind functions to. On keyboards that lack such a key, the ASCII ESCape key is used as the first of a two-key sequence to represent the same thing. This is written M-a, meaning to hold down the meta key while striking the a, or to type the two characters ESCape a in succession.

However, even $33 + 128$ functions are not enough, so one or more key sequences are used as prefixes to named commands, or to get another 128 command-letter possibilities. Thus, the command M-x sort-lines runs a function to sort lines in the current region (a subset of the buffer), and for editing LaTeX text, C-c f generates a `\footnote{}`.

All Emacs key bindings are customizable, and importantly, that customization can be *unique* to each edit buffer. To reduce confusion, the commonest functions, those attached to most control and meta keys, are rarely rebound, and even when their bindings are changed, the functionality provided is usually similar. There is a good analogy here with TeX's catcodes, which also permit characters to take on new meanings.

Keyboards on some terminals, the IBM PC, and workstations have additional keys with cursor direction arrows, and labels like Help, Home, and Page Down. These keys usually generate multi-character escape sequences, or special scan codes. Emacs' key binding model is general enough to deal with these special codes, allowing functions to be bound to them. Novices often find such keys convenient, but fast touch typists prefer the conventional key bindings because they need not move their hands from the standard typing position.

A collection of key-function bindings suitable for a particular task is known as an *editing mode*. A mode can be selected by name, such as in the command M-x LaTeX-mode, or it can be set automatically according to the extension of the file name that has just been selected for editing. The connection between file extensions and Emacs editing modes is of course user-customizable.

Most Emacs functions can be given arguments when they are invoked. In the simplest case, a numeric argument says to do the command n times. The function universal-argument is bound to C-u, so typing C-u -31 produces an argument of -31. If no digits or minus sign follow, an argument of 4 is provided, and if the command is used more than once in succession, the argument is multiplied by four each time: C-u C-u C-u x will insert 64 x's into the buffer. Multiple C-u keystrokes are often used to speed up cursor movement.

Some functions just use the presence or absence of an argument to vary their behavior slightly, like LaTeX's starred and unstarred commands. Functions that expect string arguments prompt for them.

Like TeX and the C language, and *unlike* Common Lisp and most other programming languages, GNU Emacs Lisp is *case sensitive*. If you type M-x latex-mode instead of M-x LaTeX-mode, then depending on the order of library loading, you might get only the limited support for LaTeX provided in the standard GNU Emacs tex-mode. Had the prefix latex- not already been usurped by tex-mode, I would have employed it myself, because it is easier to type than LaTeX-. Perhaps that can be changed in the future. In any event, most LaTeX-mode functions have key bindings, so only rarely do you have to type their names.

In any text editor, it is always possible to make a serious editing mistake that damages your work. Emacs offers several features that help you recover. It has an auto-save-mode that causes backup copies of the edit buffers to be saved in the file system. You can select how many file generations to save, and how often auto-saving occurs. Emacs catches fatal errors, like loss of a phone line or network connection, and saves your work. On request, it can show you what you have recently typed, so you can perhaps figure out what you did wrong. It also provides an extensive *undo facility* that lets you revert to an earlier stage of editing; if you accidentally undo too much, you can undo your undo commands, retracing your steps to reach the point you want.

4 Emacs documentation

TeXinfo, and its more recent relative, LaTeXinfo, are greatly-restricted TeX and LaTeX subsets. Files in these formats can be typeset by TeX or LaTeX, or converted with the `makeinfo` program, or the Emacs texinfo-format-buffer function, into a file in a special format that allows easy navigation by the Emacs `info` system.

`info` is a mature implementation of a *hypertext* system, subject to the constraint that text must be viewable using only the 95 printing ASCII characters.

Simple commands allow ‘swinging through the branches’ of the `info` tree, selecting topics from menus, searching forward and backward, moving up, down, and sideways through the branches, jumping temporarily into cross-references or to arbitrary nodes by name, and retracing one’s node path which `info` always remembers. Since the `info` text is always viewed in an Emacs buffer, which might be visible in one of several windows on the screen, documentation can be read (and edited) while you are inside Emacs. You can enter and leave `info` at will. Emacs always remembers where you were, so reentry brings you back to the exact spot in the documentation that you were reading before.

5 Learning an Emacs editing mode

When faced with a new editing mode (i.e. set of key-function bindings), an Emacs user wants to be able to find out how to use it. This can be done in several ways:

- Sometimes, there is printed documentation of the kind you will see later in this article. If that documentation is prepared in TEXinfo format, it can be accessible online as well as in the Emacs `info` system.
- Most editing modes provide a brief summary of their usage in response to the command M-x describe-mode, conventionally bound to C-h m. C-h is the Emacs Help key; the keyboard also generates C-h when the BackSpace key is pressed. The output of this command in LaTeX-mode is shown in Figure 1 and Figure 2 on page 95.
- Documentation of individual commands or keys can be requested: C-h d (describe-function) LaTeX-index and C-h k (describe-key) C-c x both display the text shown in Figure 3 on page 95.
- C-h w (where-is) tells what the key binding of a named function is.
- The command M-x apropos searches the list of loaded functions to find all those with a particular phrase in their names.
- The function describe-bindings will display in a separate buffer a list of all of the local (buffer-specific) and global (all buffers) key bindings. The local bindings for LaTeX-mode produced by that function are shown in Figure 4 and Figure 5 on page 96.

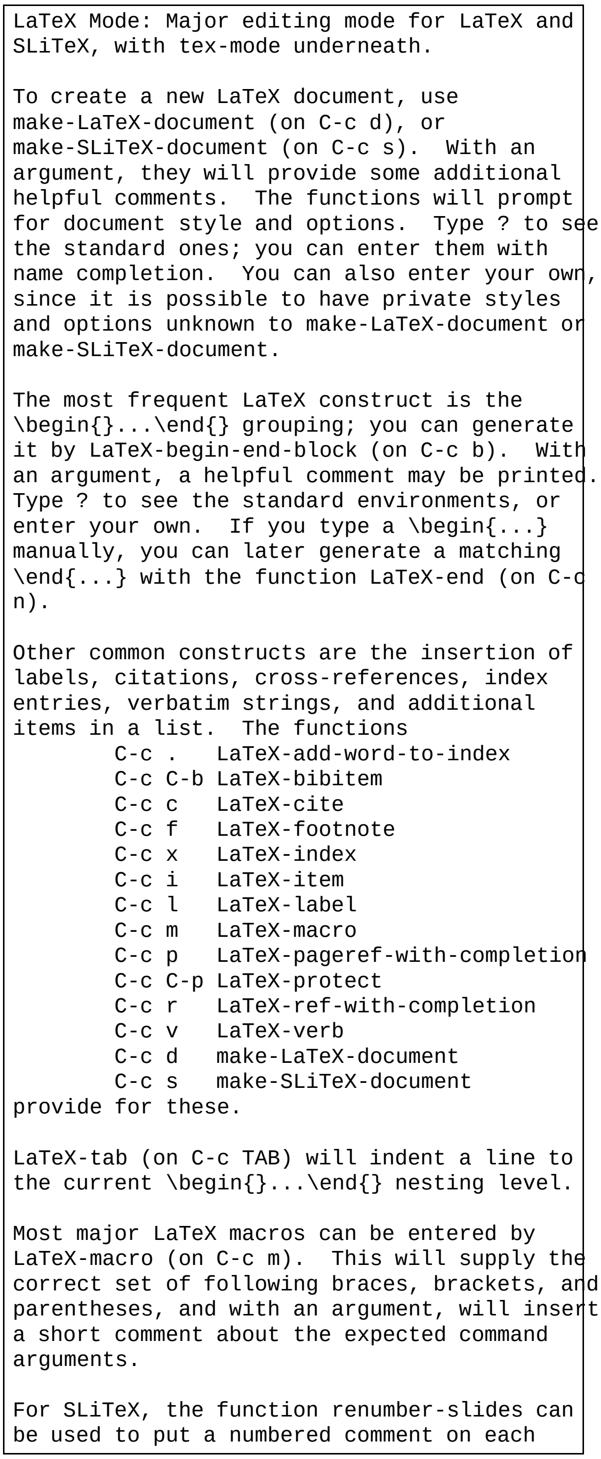
6 LATEX support

Now that you understand how Emacs works, we can move on to a discussion of the LATEX editing support.

6.1 Preliminaries

The first job is to arrange for LaTeX-mode to be selected automatically when required.

Because the `latex.el` file of Lisp code is not yet a standard part of the GNU Emacs distribution, its as-



```
LaTeX Mode: Major editing mode for LaTeX and
SLiTeX, with tex-mode underneath.

To create a new LaTeX document, use
make-LaTeX-document (on C-c d), or
make-SLiTeX-document (on C-c s).  With an
argument, they will provide some additional
helpful comments.  The functions will prompt
for document style and options.  Type ? to see
the standard ones; you can enter them with
name completion.  You can also enter your own,
since it is possible to have private styles
and options unknown to make-LaTeX-document or
make-SLiTeX-document.

The most frequent LaTeX construct is the
\begin{}...\end{} grouping; you can generate
it by LaTeX-begin-end-block (on C-c b).  With
an argument, a helpful comment may be printed.
Type ? to see the standard environments, or
enter your own.  If you type a \begin{...}
manually, you can later generate a matching
\end{...} with the function LaTeX-end (on C-c
n).

Other common constructs are the insertion of
labels, citations, cross-references, index
entries, verbatim strings, and additional
items in a list.  The functions
        C-c .   LaTeX-add-word-to-index
        C-c C-b LaTeX-bibitem
        C-c c   LaTeX-cite
        C-c f   LaTeX-footnote
        C-c x   LaTeX-index
        C-c i   LaTeX-item
        C-c l   LaTeX-label
        C-c m   LaTeX-macro
        C-c p   LaTeX-pageref-with-completion
        C-c C-p LaTeX-protect
        C-c r   LaTeX-ref-with-completion
        C-c v   LaTeX-verb
        C-c d   make-LaTeX-document
        C-c s   make-SLiTeX-document
provide for these.

LaTeX-tab (on C-c TAB) will indent a line to
the current \begin{}...\end{} nesting level.

Most major LaTeX macros can be entered by
LaTeX-macro (on C-c m).  This will supply the
correct set of following braces, brackets, and
parentheses, and with an argument, will insert
a short comment about the expected command
arguments.

For SLiTeX, the function renumber-slides can
be used to put a numbered comment on each
```

Figure 1: Online summary of LATEX mode.

```
slide environment for handy reference in the
\onlynotes{...} and \onlyslides{...} commands.

You can move over \begin{} ... \end{} groups
with LaTeX-to-begin (on C-c a) and
LaTeX-to-end (on C-c e).

Insertion of paired angle brackets, braces,
square brackets, and parentheses is provided
by
        C-c <  LaTeX-insert-angles
        C-c {  LaTeX-insert-braces
        C-c [  LaTeX-insert-brackets
        C-c (  LaTeX-insert-parentheses
With an argument, the last three insert
backslash prefixes for literal braces, math
mode, and displaymath mode.

Unbalanced character pairs are found by
        C-c >  LaTeX-check-angle-balance
        C-c }  LaTeX-check-brace-balance
        C-c ]  LaTeX-check-bracket-balance
        C-c $  LaTeX-check-dollar-balance
        C-c )  LaTeX-check-parenthesis-balance

You can test for undefined labels with
check-LaTeX-labels, display the labels and
their line numbers with show-LaTeX-labels, and
fetch labels from an .aux file with
update-LaTeX-labels.

Environment nesting errors can be caught by
check-LaTeX-nesting.

Use indent-LaTeX-begin-end-groups to get
environments nested for improved visibility.

Use LaTeX-comment (on C-c %) to comment out
the current paragraph (no arg) or region
(arg); undo with M-X undo (C-_).  LaTeX-
uncomment (on C-c u) is the inverse function.

One or more words can be set in any standard
LaTeX font using the commands
        C-c C-c b       LaTeX-font-bf
        C-c C-c e       LaTeX-font-em
        C-c C-c i       LaTeX-font-it
        C-c C-c r       LaTeX-font-rm
        C-c C-c u       LaTeX-font-sc
        C-c C-c f       LaTeX-font-sf
        C-c C-c s       LaTeX-font-sl
        C-c C-c t       LaTeX-font-tt

Please report bugs, comments, and enhancements
by e-mail to
        beebe@math.utah.edu (Internet)
LaTeX-gripe (on C-c g) makes this easier.
```

Figure 2: More about LaTeX mode.

```
LaTeX-index:
Insert \index{ ... } at point (on C-c x).

If a prefix argument is given, the string is
prompted for, and inserted both as text and as
an index entry, e.g. gnats and
gnus\index{gnats and gnus}.  All horizontal
space preceding \index is removed to prevent
an intervening page break causing an
off-by-one page number error.

If LaTeX-index-start-with-newline is non-nil,
insert a percent to start a comment, then put
\index at the start of a following new line.
If LaTeX-index-end-with-newline is non-nil,
follow the entry with a new line.
```

Figure 3: Online help for LaTeX-index.

sociation with particular file extensions is unknown to Emacs. To remedy that, the Lisp code in Figure 6 on page 96 should be inserted into the `.emacs` file in your login directory; that file is processed each time Emacs starts up. The load command there tells where `latex.el` is found; it may need adjustment for your system.

The peculiar if commands attach LaTeX-mode to the *auto-mode-alist* variable, which is a list of pairs of file extensions and their mode names; the attachment is omitted if the file extensions are already in the list.

Once this startup code has been executed, Emacs will automatically select LaTeX-mode when you visit files with extensions `.ltx` (LaTeX document), `.stx` (SLiTeX document), or `.sty` (LaTeX style). Others can be easily added according to taste. File extension `.tex` is already attached to GNU Emacs tex-mode which comes with the system, so different extensions are needed for different editing modes.

If you don't have these associations of file extensions with editing modes in effect, you can always select the mode manually with the command M-x LaTeX-mode.

Experienced Emacs users may wonder why the autoload facility is not used instead of the explicit load command in the startup file; the reason lies in a conflict with the standard tex-mode. I expect that conflict will be removed when `latex.el` becomes part of the standard GNU Emacs distribution. A modification of `tex-mode.el` that eliminates all LaTeX-specific code has been prepared locally, and that new version removes the incompatibility with `latex.el`, allowing autoload to be used.

In GNU Emacs Lisp, named functions, and variables defined outside functions, are known globally. There is no provision for file scoping or mode scoping to limit name visibility. Consequently, to avoid collision with the thousands of names from other libraries, each library usually includes a distinctive phrase in its global names. Naturally enough, LaTeX and SLiTeX

```
Local Bindings:
key            binding
---            -------

_              LaTeX-subscript
ˆ              LaTeX-superscript
.              LaTeX-dot
,              LaTeX-comma
$              LaTeX-dollar
ESC            Prefix Command
LFD            newline-and-indent
C-c            Prefix Command
"              LaTeX-insert-quote

C-c ,          LaTeX-set-indentation
C-c C-c        Prefix Command
C-c =          LaTeX-equation
C-c }          LaTeX-check-brace-balance
C-c {          LaTeX-insert-braces
C-c x          LaTeX-index
C-c v          LaTeX-verb
C-c u          LaTeX-uncomment
C-c s          make-SLiTeX-document
C-c r          LaTeX-ref-with-completion
C-c p          LaTeX-pageref-with-completion
C-c n          LaTeX-end
C-c m          LaTeX-macro
C-c l          LaTeX-label
C-c i          LaTeX-item
C-c g          LaTeX-gripe
C-c f          LaTeX-footnote
C-c e          LaTeX-to-end
C-c d          make-LaTeX-document
C-c c          LaTeX-cite
C-c b          LaTeX-begin-end-block
C-c a          LaTeX-to-begin
C-c 0          renumber-slides
C-c ]          LaTeX-check-bracket-balance
C-c [          LaTeX-insert-brackets
C-c .          LaTeX-add-word-to-index
C-c )          LaTeX-check-parenthesis-balance
C-c (          LaTeX-insert-parentheses
C-c %          LaTeX-comment
C-c $          LaTeX-check-dollar-balance
C-c >          LaTeX-check-angle-balance
C-c <          LaTeX-insert-angles
C-c C-p        LaTeX-protect
C-c C-n        LaTeX-news
C-c TAB        LaTeX-tab
C-c C-b        LaTeX-bibitem

ESC )          forward-list
ESC (          backward-up-list
```

Figure 4: Local bindings in LATEX mode.

```
C-c C-c t      LaTeX-font-tt
C-c C-c s      LaTeX-font-sl
C-c C-c f      LaTeX-font-sf
C-c C-c u      LaTeX-font-sc
C-c C-c r      LaTeX-font-rm
C-c C-c i      LaTeX-font-it
C-c C-c e      LaTeX-font-em
C-c C-c b      LaTeX-font-bf
C-c C-c }      LaTeX-check-brace-balance
C-c C-c ]      LaTeX-check-bracket-balance
C-c C-c )      LaTeX-check-parenthesis-balance
C-c C-c $      LaTeX-check-dollar-balance
C-c C-c =      LaTeX-displaymath
```

Figure 5: More local bindings.

```
;=============================================
; Add mode for .ltx files if not there already
; for most people: (load-library "latex.el")
; for my private version:
(load "˜/emacs/latex" t t nil)
(if (not (assoc "\\.ltx$" auto-mode-alist))
    (setq auto-mode-alist
          (cons (cons "\\.ltx$" ’LaTeX-mode)
                auto-mode-alist)))
(if (not (assoc "\\.stx$" auto-mode-alist))
    (setq auto-mode-alist
          (cons (cons "\\.stx$" ’LaTeX-mode)
                auto-mode-alist)))
(if (not (assoc "\\.sty$" auto-mode-alist))
    (setq auto-mode-alist
          (cons (cons "\\.sty$" ’LaTeX-mode)
                auto-mode-alist)))

;=============================================
; Private letter support
(load "˜/emacs/letter" t t nil)
```

Figure 6: `.emacs` file additions.

are the phrases chosen in `latex.el`. They are always used as prefixes in variable names, and usually also in function names, although a few functions have embedded phrases for better readability (e.g. show-LaTeX-labels).

6.2 Creating a new LaTeX file

When you visit a new LaTeX file, Emacs starts up with an empty buffer. The mode line at the bottom of the screen will look something like

```
---Emacs: foo.ltx (LaTeX Abbrev Fill)----All---
```

The first thing you want to do is get a standard template of a LaTeX file inserted; you do this by typing C-c d (make-LaTeX-document). This function will first prompt you with

```
Document style:
```

to which you can respond either with a complete style file name, or with the first few characters of a standard style, followed by a space to request Emacs to complete the name, and a RETurn to accept the completion. If you are unsure of what to supply, just type a query (?); Emacs will respond with something like

```
Possible completions are:
aaai                                   amsart
amsbook                                amstex
aps                                    article
book                                   deproc
letter                                 ltugproc
refman                                 report
siam                                   slides
tugboat                                ucthesis
uoftoronto                             usafmemo
uuthesis                               xxxslides
```

Completion can also be requested on partial names: the input a? in the above command will respond with

```
Possible completions are:
aaai                                   amsart
amsbook                                amstex
aps                                    article
```

Suppose you type `article` or `ar␣`, followed by a RETurn. Then Emacs will respond with

```
Document option (RET when done):
```

to which you can then supply an option name followed by a RETurn. As before, a query will show you what is available:

```
Possible completions are:
11pt                                   12pt
a4                                     a4wide
acm                                    acronym
agugrl                                 agujgr
alltt                                  amsbsy
amscd                                  amsfonts
amssymb                                amssymbols
amstext                                apalike
archmemo                               array
bezier                                 biihead
boxedmini                              changebar
chapterbib                             cite
clsc                                   clscmemo
concrete                               cropmark
ctagsplt                               cyrillic
dblspace                               doublespace
draft                                  drafthead
drop                                   dvidoc
eqsecnum                               espo
fleqn                                  format
fullpage                               geophysics
german                                 gnuindex
ifthen                                 intlim
ist21                                  leqno
listing                                longtab
ltugboat                               makeidx
math                                   merge
mfr                                    mitthesis
moretext                               natsci
nl                                     nonamelm
nopageno                               nosumlim
openbib                                pandora
preprint                               proc
psmavg                                 psmbkm
psmncs                                 psmpal
psmtim                                 remark
resume                                 revtex
righttag                               sc21
sc21-wg1                               sfwmac
showidx                                showlabels
slem                                   spacecites
supertab                               suthesis
tablisting                             tapeseal
texindex                               texnames
tgrind                                 theorem
thp                                    threepart
titlepage                              trademark
troffman                               troffms
twocolumn                              twoside
underline                              uofutah
vdm                                    verbatim
```

You can keep on selecting options until you finally just type a bare RETurn, at which point make-LaTeX-document completes, and you have a buffer that looks like this:

```
% -*-latex-*-
% Document name: /u/beebe/foo.ltx
% Creator: Nelson Beebe [beebe@math.utah.edu]
% Creation Date: Sun Jul 28 08:44:46 1991
\documentstyle[11pt,makeidx]{article}
\newcommand{\X}[1]{{#1}\index{{#1}}}
\begin{document}
□
\end{document}
```

The cursor, represented by □, is left positioned on the line following the `\begin{document}`, ready for text entry.

The initial comment with the tag `-*-latex-*-` is a special one. It requests an automatic mode selection when the file is freshly visited in Emacs, overriding any mode that might otherwise be selected based on the file name extension. latex-mode is made equivalent to LaTeX-mode in `latex.el` so that the comment tag can be written in lower-case, as is conventional.

The `% Creator:` comment information is obtained from Emacs' operating system interface, which in turn fetches the user's personal name and login name from system authorization files. On networked systems, the hostname is included as well, creating a valid electronic mail address.

Had you given make-LaTeX-document a numeric argument, it would have been somewhat more verbose:

```
% -*-latex-*-
% Document name: /u/beebe/tex/tugboat/foo.ltx
% Creator: Nelson Beebe [beebe@math.utah.edu]
% Creation Date: Sun Jul 28 08:46:33 1991
%----------------------------------------------
% EVERYTHING TO THE RIGHT OF A  %  IS A REMARK
% TO YOU AND IS IGNORED BY LaTeX.
%
% WARNING!  DO NOT TYPE ANY OF THE FOLLOWING 10
% CHARACTERS AS NORMAL TEXT CHARACTERS:
%       &   $   #   %   _    {   }   ˆ   ˜   \
%
% The following seven are printed by typing a
% backslash in front of them:
%       $  &  #  %  _  {  and  }.
%----------------------------------------------
\documentstyle[11pt,makeidx]{article}
\newcommand{\X}[1]{{#1}\index{{#1}}}
\begin{document}
□
\end{document}
```

The extra commentary is a useful reminder for beginning LATEX users. Several other commands in LaTeX-mode use the presence of a numeric argument to supply additional helpful commentary.

The \X command is one I have found helpful for preparing indexes; typing \X{gnats and gnus} will put that phrase in both the document and the index. Further discussion is given in the **Indexing** section beginning on page 103.

The lists of standard styles and options are embedded in the latex.el code. They cannot be determined automatically from the file system for several reasons:

- Some options (e.g. 11pt) do not have a corresponding style file.
- Some style files do not represent a valid option (e.g. art10.sty).
- Some style files found in the normal *TEXINPUTS* search path belong to other systems, such as AMS-TEX.
- It is impossible to distinguish from the .sty file extension whether the file represents a major document style, or merely an option that modifies a major style.

It is regrettable that these difficulties exist, because they confuse users, as well as programmers of code like LaTeX-mode. It is often difficult to tell from its contents whether a particular file is a major style, or just a document option. It would have been wiser in the initial LATEX design to have chosen distinctive file extensions, and for each style and option to be associated with a unique file.

The present collection of twenty available styles and over one hundred options is daunting for a novice, particularly since many options can only be used with certain styles. Eventually I may find a satisfactory way to deal with this, and offer a better presentation of choices in the completion lists.

The style and option lists can be easily extended; see **Customizing lists** on page 106 for details.

6.3 LATEX macros

Every one of the 589 LATEX macros in the index to the *LATEX User's Guide and Reference Manual* [9] is known to the function LaTeX-macro (on C-c m), and as before, Emacs command completion is available with the query (?) and space (␣) characters. Thus, you can type

- C-c m longr␣ RETurn to get \longrightarrow;
- C-c m em RETurn to get
  ```
  {\em □
  \/}
  ```
  with the cursor positioned ready for you to enter some text;
- C-u C-c m newcom␣ RETurn to get
  ```
  \newcommand{□
  }[]{} % {\cmd}[n]{def} define new command
        % with n arguments
  ```
- C-c m e? to find the names of macros that begin with e:
  ```
  Possible completions are:
  ell                              em
  emptyset                         encl
  end                              epsilon
  equiv                            eta
  evensidemargin                   exists
  exp                              extracolsep
  ```

When the cursor is left after an open brace in a generated command, the closing brace is by default placed at the start of the next line so that you have a clean line to type on. This reduces visually-distracting screen updates, but means that you will need to delete the end-of-line character when you finish typing the argument. If you don't like this behavior, you can set the Emacs variable *LaTeX-newline-after-opening-delimiter* to a nil value, usually in your .emacs startup file:

```
(setq LaTeX-mode-hook
 '(lambda ()
    (setq LaTeX-newline-after-opening-delimiter
          nil)
    )
 )
```

The variable *LaTeX-newline-after-closing-delimiter* is used similarly for brace pairs inserted on their own, or after certain macros.

A *mode hook* is Lisp code to be executed whenever the corresponding mode function is executed. The hook is called just before the mode function returns, so it can override anything that the function did. The peculiar lambda () is the Lisp way of declaring a nameless function. The single quote prefixing it means that *LaTeX-mode-hook* will be assigned the function itself, and not the value returned by the function.

In GNU Emacs, you can execute Lisp code in a file through M-x load-file and M-x load-library commands. Inline code must be executed in a buffer which is in emacs-lisp-mode. The minibuffer always has that mode, and can be temporarily accessed with the eval-expression function bound to ESCape ESCape.

If you just want to set an Emacs variable, you can also use the command M-x set-variable; it supports completion on the variable name, and prompts for the value.

The availability of all of the LaTeX macros with LaTeX-macro reduces the likelihood of spelling mistakes, and the command completion usually means that only a few leading characters need to be typed. The complete macro name is entered in the buffer: `\abovedisplayshortskip` is much more readable than a cryptic `\adss` that LaTeX users without such editing support might be inclined to define as a private macro to save typing effort.

If you have frequently-used private macros, there is a convenient way to make them known to LaTeX-mode. See the section **Customizing lists** on page 106 for details.

6.4 LaTeX environments

An important concept in LaTeX is the notion of *environments*, which are marked by surrounding `\begin{...}` and `\end{...}` commands. The braced tag must be repeated, and if the tag name is long, spelling errors are more likely. The function LaTeX-begin-end-block on C-c b generates the command pair, indented according to the current nesting level, and leaves the cursor on the line between them. The indentation is controlled by the value of the Emacs variable *LaTeX-begin-end-indentation*; the default is two spaces.

If the variable *LaTeX-space-after-begin-end* is non-nil, a space will be inserted between the keyword and the following opening brace. The default value is nil. This formatting style is discouraged, because it cannot be applied uniformly; it must be suppressed for the `verbatim` environment. The code in `latex.el` knows about this vagary, and always omits the space for that environment.

The keystrokes C-c b quote RETurn C-c b ite␣ RETurn C-c b verb␣ RETurn produce

```
  \begin{quote}
    \begin{itemize}
      \item
\begin{verbatim}
□
\end{verbatim}
      \item
      \item
      \item
    \end{itemize}
  \end{quote}
```

with the cursor inside the `verbatim` environment. Notice that the normal indentation is automatically suppressed for that environment; the command knows that leading spaces in front of `\end{verbatim}` would otherwise generate an unwanted blank line in the typeset output.

If you have private environments, you can still use C-c b; you just have to type the complete environment name. If you use some private ones frequently in different documents, it is possible to extend the list of environments known to LaTeX-mode. See the section **Customizing lists** on page 106 for details.

In those environments that have `\item` commands, four are generated by default; that number can be changed by modifying the value of the Emacs variable *LaTeX-item-count*. Their indentation is defined by the variable *LaTeX-item-indentation*; the default is two spaces. Additional indentation of text under a `\item` command is set by the variable *LaTeX-item-info-indent*; the default is six spaces.

You can generate additional `\item` commands as needed with the function LaTeX-item on C-c i. With an argument, it supplies the alternate form `\item[]` instead.

If you forgot about the C-c b command, and manually typed a `\begin{conjecture}`, you can get the matching `\end{conjecture}` generated automatically by typing C-c n (LaTeX-end). That is also sometimes helpful when you have forgotten what environment you are in: you can generate the `\end{...}`, and then kill it.

You can type C-c a (LaTeX-to-begin) to move backward to the enclosing `\begin`; with an argument, you can move backward over a specified number of `\begin{...}`s. If the argument is negative, you move forward over `\end{...}`s instead. Usually, you would use C-c e (LaTeX-to-end) instead to move forward over `\end{...}`s; that command handles numeric arguments too, and reverses direction for negative arguments.

These two movement commands are analogous to the standard Emacs commands for moving backward and forward in sentences, normally bound to M-a and M-e. They also require that the `\begin` and `\end` macros be preceded on their lines only by optional whitespace, and immediately followed by open braces. This helps to enforce the discipline of keeping the `\begin{...}` ... `\end{...}` grouping properly nested and readable, and also avoids having to deal with more complex input parsing.

6.5 Common commands

Some LaTeX macros are used so often that it is desirable to have them bound directly to keys, instead of having to type their first few characters with LaTeX-macro as described earlier. You can find a list of these convenience functions in the fourth paragraph of Figure 1 on page 94.

For example, typing C-c l (that's the letter 'l') generates

```
~\label{□
}
```

at the cursor position, leaving the cursor after the open brace.

6.6 Math mode

TeX's math mode uses paired dollar signs to delimit the math text. Single dollar signs mark inline math, and doubled ones mark display math. If you forget to type a closing dollar sign, or type the wrong number of dollar signs, TeX will complain bitterly.

The start and finish of these math regions are indistinguishable, making it hard for simple programs (and sometimes, readers) to distinguish the 'inside' from the 'outside'.

Consequently, LaTeX additionally supports two bracketing forms: `\(` ... `\)` for `$ ... $`, and `\[` ... `\]` for `$$ ... $$`, as well as more readable forms

```
\begin{math}
...
\end{math}

\begin{displaymath}
...
\end{displaymath}
```

for the single and double dollar sign pairs, respectively. All of these have unique opening and closing sequences.

LaTeX-mode supports all of these forms. The variants selected are determined by the variables *LaTeX-math-option* and *LaTeX-displaymath-option*. Values of 0, 1, and 2 select the dollar, backslashed delimiter, and `\begin{...}` ... `\end{...}` environment forms respectively; all other values are equivalent to 0. The default values select the single dollar and `displaymath` forms.

Normally, when you type a dollar sign, you get an inline math environment. If the dollar sign happens to be at the beginning of a line, it expands instead into a display environment. On the other hand, if the preceding character was a backslash, you just get a plain dollar sign.

Spacing around the inserted text is controlled by two more variables. If *LaTeX-newline-after-opening-delimiter* is non-nil (the default), the opening delimiter is followed by a newline, with the cursor positioned immediately after the delimiter for the inline case, and on a new line for the display form. If *LaTeX-newline-after-closing-delimiter* is non-nil (the default), the closing delimiter is followed by a newline.

This pleasantness is our first example of having an alternate function bound to a normal printing character. Instead of running the usual self-insert-command attached to printing characters, $ is bound to LaTeX-dollar which has been programmed to deal with the three different meanings of the dollar sign. Few editors outside the Emacs family could do this; the critical feature is that *any character* can run *any function*.

What if you really just wanted a single plain dollar sign, with no preceding backslash? Well, that is easy. In Emacs, any character can be *quoted*, which causes it to be inserted directly into the buffer; the quoted-insert function is normally bound to C-q, but that too can be changed. Thus, C-q $ will get you a literal dollar sign, if that is what you really want.

The function LaTeX-equation on C-c = generates

```
  \begin{equation}
□
  \end{equation}
```

with the cursor between them. LaTeX-displaymath on C-c C-c = inserts

```
  \begin{displaymath}
□
  \end{displaymath}
```

All math mode environments (`array`, `eqnarray*`, `eqnarray`, `math`, and `theorem`) can be also generated by LaTeX-begin-end-block on C-c b, and additional ones can be easily added; see **Customizing lists** on page 106.

LaTeX-subscript on _ and LaTeX-superscript on ^ generate braced subscript and superscript sequences, leaving the cursor between the braces [16].

6.7 Dots, commas, and quotation marks

The dot or period (full stop, to some of you) is an overloaded character. Normally, it just means end of sentence. Sometimes, it ends an abbreviation in the middle of a sentence. It can also be a decimal point, as in 3.14159. Still other times, it appears in ellipses, as in $\cdots$ (`\cdots`), $\ddots$ (`\ddots`), and $\ldots$ (`\ldots`).

In typography, these uses are distinguished by small changes in spacing after the dots. That is why TeX provides control sequences for the ellipses, and the LaTeX book [9, p. 14] recommends that you type a `\` sequence after a period that does not end a sentence.

TeX assumes that a period following an upper-case letter does not end a sentence (it might be someone's initial) [8, pp. 74, 311], and follows it with a smaller amount of space. The *LaTeX User's Guide and Reference Manual* recommends insertion of `\@` before a sentence-ending dot that follows an upper-case letter in order to get additional space after the dot.

LaTeX-mode handles these intricacies by binding dot to the function LaTeX-dot. That function examines what is immediately before the dot in the buffer, and takes one of several actions:

- If the dot ends an ellipsis, the three dots are replaced by `\ldots{}`.
- If the dot is preceded by an italic correction at group end, `\/}`, the italic correction is removed.
- If the dot ends an abbreviation, like *e.g.* or *i.e.*, it inserts a `\ ` command or a comma, depending on whether the variable *LaTeX-comma-after-dotted-abbreviation* is nil or non-nil. Traditionally, such abbreviations are followed by a comma, but modern tendency is to omit the comma [14, p. 84], although Knuth disagrees [8, p. 74].
- If the previous characters are abbreviations like *Fig.*, *cf.*, *vs.*, and *resp.* [8, p. 74] [9, p. 18] that should be tied to the following word with a single unbreakable space, the function inserts a tilde to mark the tie.
- If the dot follows two or more upper-case letters, the dot is prefixed by `\@`.

To prevent confusion with words at end of sentence, abbreviations are not recognized if they are immediately preceded by a letter in the editing buffer.

The abbreviations known to LaTeX-dot are defined by the variable *LaTeX-standard-dotted-abbreviations*. More can be added by a suitable definition of the variable *LaTeX-extra-dotted-abbreviations*, such as this example:

```
(setq LaTeX-extra-dotted-abbreviations
      '(
        ("ad. lib.")
        ("cwt.")
        ("q.e.d.")
        )
      )
```

See **Customizing lists** on page 106 for more details.

The abbreviations for which ties are supplied are similarly defined by the variables *LaTeX-standard-tied-abbreviations* and *LaTeX-extra-tied-abbreviations*.

LaTeX-dot cannot tell whether a dot following an upper-case letter is a sentence-ending one or not, so it only inserts `\@` before a dot if there are *two or more* preceding upper-case letters. It is up to the user to remember to insert `\@` in the exceptional case of a single final upper-case letter.

What about a dot after a macro that expands to a word ending in an upper-case letter, like TeX itself? No `\@` is necessary there, because TeX treats that dot as a normal end of sentence.

These actions cover most of the common cases, but of course, for abbreviations, cannot be omniscient. Dictionaries that I consulted listed over a thousand abbreviations in use in English. Fewer than a dozen of the commonest ones are included in *LaTeX-standard-dotted-abbreviations*, but customization of *LaTeX-extra-dotted-abbreviations* provides a way to add new ones.

Emacs' quoted-insert function saves the day when you want to suppress all of this wizardry; just type C-q . to get a literal dot inserted.

LaTeX-comma bound to comma removes any immediately preceding italic correction, and then inserts a comma.

Italic corrections are somewhat of a nuisance, since you are supposed to have them after italicized text, *except* a period or a comma, which are so low that it doesn't matter if the character to their immediate left leans over them. Those two exceptions make it hard to write a TeX macro to determine whether the correction is needed or not. I'd rather let Emacs remember for me, so I programmed it to do so.

Good typography uses different opening and closing quotation marks; in TeX, you get "quoted text" by typing ``` ``quoted text'' ```. Because many people are accustomed to using the double quotation mark character, ", found on computer keyboards and typewriters, the function LaTeX-insert-quote is bound to that character. It generates the correct paired opening grave accents, ``` `` ```, or closing apostrophes, `''`, as appropriate: the input `"quoted text"` expands to ``` ``quoted text'' ``` in the edit buffer. With an argument, the function inserts a bare double quotation mark; you can get the same effect from quoted-insert by typing C-q ".

6.8 Font changes

The LaTeX-macro function on C-c m lets you easily generate font changes to any of LaTeX's eight standard styles, including supplying the italic correction in the three leaning styles (`\em`, `\it`, and `\sl`).

But what if you are revising a document, and decide to add a font change? Well, you could use C-c m to do it, but then you'd have to move the closing brace, and any preceding italic correction.

LaTeX-mode provides a better solution: customized functions *wrap* a font change around the current word, or if a numeric argument is given, around that many words starting from the current one. These functions are listed near the bottom of Figure 2 on page 95. They are almost the only functions that require a second prefix character; C-c C-c e (LaTeX-font-em) is easy to type, and retains mnemonic significance. These functions handle the italic correction properly; leaning fonts get it unless the character following the last word is a period or a comma.

6.9 Comments

TeX provides an inline comment mechanism with the percent sign. It discards text from that character to the end of the line, plus *all leading space on the next line up to, but not including, its end-of-line character*.

Sometimes, however, you want to temporarily suppress typesetting of a part of a document without actually removing it from the file. One way is to insert leading percents on each line of the region to be suppressed, but that is tedious to do manually. A second way is to use LaTeX's `ifthen` style option which added loops and conditionals, but appeared after the *LaTeX User's Guide and Reference Manual* was written, and so is not widely known. A third way is to define a LaTeX macro like

```
\newcommand{\comment}[1]{}
```

However, that won't work if the argument contains paragraph breaks. You could drop down to low-level TeX and write

```
\long\def\comment#1{}
```

but LaTeX users are not supposed to do that, except in style files. Also, both of these macros risk overflowing TeX's input buffer, since the complete argument might be rather long, even if it is subsequently discarded.

I often wish that LaTeX had provided a standard `comment` environment to do this; it could be implemented much like `verbatim`, except that it must support properly nested `comment` environments. Rainer Schöpf has implemented a version of a `comment` environment in a new `verbatim` implementation for LaTeX [13], although nested comments are not yet supported.

The solution provided by LaTeX-mode is the function LaTeX-comment on C-c % which implements the first choice above. By default, it comments out the current paragraph, or with an argument, the current region. The comment prefix inserted at the beginning of each line is defined by the variable *LaTeX-comment-prefix*; its default value is `%%: `. A unique prefix is desirable to distinguish such lines from ordinary comments. You should avoid using any regular-expression matching characters in it.

You can remove the comments inserted by LaTeX-comment with the function LaTeX-uncomment on C-c u. It should normally be used with an argument to act on a region, instead of a paragraph. The reason for this is that paragraphs are recognized according to the regular expression in the Emacs variable *paragraph-separate*; the default setting is such that a line containing only a comment ends a paragraph. Consequently, LaTeX-uncomment will do nothing because its 'paragraph' is always empty. Invoking it with an argument to process a marked region solves the problem.

6.10 Mismatched delimiters

TeX gets very unhappy when it finds mismatched braces, and in complex documents, it can sometimes be difficult to find them. The job is not made easier on low-resolution screen displays where braces and parentheses are virtually indistinguishable. Finding such mismatches is a job for a computer, not a human, and LaTeX-mode offers you help.

Braces are not the only things you might want assistance with. Checking for mismatched parentheses, square brackets, and angle brackets is also supported. These normally insert themselves when you type them, but LaTeX-mode has commands that insert pairs: typing C-c { gets you a pair of open and close braces, and similarly for C-c [, C-c (, and C-c <. These are the default bindings of the functions LaTeX-insert-braces, LaTeX-insert-brackets, LaTeX-insert-parentheses, and LaTeX-insert-angles.

If you get in the habit of using these instead of typing the delimiter pair yourself, you will rarely have mismatch problems. Emacs itself has native delimiter matching support: as you type a close delimiter, the cursor flashes briefly to the matching open delimiter.

But what if you didn't follow my advice, typed a lot of braces yourself, and then when TeX complained about unbalanced braces, you couldn't find the errors? Well, the function LaTeX-check-brace-balance bound to C-c } will scan the buffer from the current point to the end, checking for unbalanced braces, and brace pairs separated by unusually long strings, but ignoring backslashed braces. It complains if it finds a paragraph break between braces, unless you give it a numeric argument. If mismatches are detected, you are so informed, and the positions of the mismatches are remembered on Emacs' stack of marks, so you can easily get to them.

Similar functions LaTeX-check-angle-balance, LaTeX-check-bracket-balance, and LaTeX-check-parenthesis-balance are provided on C-c >, C-c], and C-c), for checking those delimiter balances. Note the key binding symmetry: C-c open-delimiter inserts the pair, and C-c close-delimiter checks the balance.

The characters < and > are likely to be used in math mode, and not require balancing. However, you might also have used them for other purposes. In such a case, you can just add matching delimiters inside comments in your math mode uses, and then C-c > will work satisfactorily.

There is also LaTeX-check-dollar-balance on C-c $. It looks for dollar pairs separated by long strings, ignoring backslashed dollars. It also complains about paragraph breaks between dollar pairs, unless it is given an argument. There is no way for a computer program that is much less complex than TeX to tell the inside of math mode from the outside, so you can confuse this function by starting it inside math mode.

The Emacs variables *LaTeX-angle-interval*, *LaTeX-brace-interval*, *LaTeX-bracket-interval*, *LaTeX-dollar-interval*, and *LaTeX-parenthesis-interval* set the limiting between-delimiter string length before a warning is raised; their default values are each set to 500 characters.

6.11 Mismatched environments

If you have a complex document where `\begin{...}` ... `\end{...}` groups have been typed manually, you can also end up with mismatched environments that are hard to find.

The command M-x check-LaTeX-nesting scans the buffer to ensure the correctness of `\begin{...}` ... `\end{...}` nesting. If, for example, your buffer contains a nesting error like this

```
  \begin{verse}
...
    \begin{quote}
...
  \end{verse}
...
    \end{quote}
```

the command will terminate with the cursor in front of the `\end{verse}` with the error message

```
This \end{verse} is matched by
preceding \begin{quote} at
mark.
```

The Emacs function exchange-point-and-mark on C-x C-x will move the cursor to the mismatching `\begin{quote}`. After fixing each such error, you can rerun the nesting check by typing C-x ESCape (Redo) until it finally reports

```
[done] -- all \begin{}-\end{}s
balance.
```

The command M-x indent-LaTeX-begin-end-groups indents `\begin{...}` ... `\end{...}` groups according to their nesting level, which helps to make the input file more readable.

You don't need these commands very often, so they do not have default key bindings.

6.12 Word abbreviations

Emacs provides a convenient word abbreviation facility that permits the automatic expansion of a string of letters when the following space or punctuation is typed. LaTeX-mode provides this by default with a set of abbreviations in a table that can be customized for personal use. The commonest ones are for macro names that are awkward to type: the input `latex␣` expands to `\LaTeX{}␣`, `tex␣` to `\TeX{} `, and so on.

Note particularly that the expansion here is to `\TeX{}`, and *not* `\TeX\␣`! There are good reasons for this:

- If you terminate macro names with `\␣` instead of with `{}`, you have to remember to do that only where a space would really be permitted; in particular, you don't want the `\␣` before punctuation. You can use `{}` consistently in both cases.
- If the `\␣` appears at the end of a line, and trailing spaces are subsequently stripped, you now have a new macro equivalent to `\^^M`, which might have a different meaning. Although both `plain.tex` and `lplain.tex` make the two macros equivalent, that might not always be the case.

  You should never count on end-of-line blanks being preserved; some e-mail systems, and some editors, may delete them. Text filling and justification as commonly used in Emacs can also remove them.
- In composite words, like `\TeX{}book`, the braces keep the parts together for text fill operations, word counting, spell checking and so forth, while the form `\TeX␣book` will not.

The form `{\TeX}` has the almost same advantages over `\TeX\␣` as `\TeX{}` does, but has one drawback: if you need to `\protect` it, then you must do so inside the braces. Use of a final empty brace pair therefore seems the best choice.

By Emacs convention, word abbreviation mode is never turned on automatically; you must explicitly toggle it on a case-by-case basis by executing the command M-x abbrev-mode. If you always want it selected in LaTeX-mode, add it to your `.emacs` file in a mode hook:

```
(setq LaTeX-mode-hook
      '(lambda ()
         (abbrev-mode 1)
         )
      )
```

If you have other things to set in that mode hook, they should go on the lines immediately before or after the `(abbrev-mode 1)` line, since there can be only one mode hook for each editing mode.

6.13 Bibliographies

Literature citations are conveniently provided for on C-c c (LaTeX-cite), which generates a `\cite{}` command with the cursor after the open brace. With an argument, it produces `\cite[]{}` instead, with the cursor between the square brackets.

There is another function, LaTeX-bibitem, on C-c C-b, which generates a `\bibitem{}` for you; with an argument, it produces `\bibitem[]{}`. Normally, you should be using a program like BibTeX [9, Appendix B] or Tib [1, 2] for preparation of bibliographies from a data base and the citations recorded in the LaTeX `.aux` file. If you find yourself using this command, you are doing the job the hard and inflexible way.

6.14 Indexing

Index preparation is a tedious job that cannot be entirely automated [3, 5, 6, 7, 10]. Nevertheless, technical documents can benefit significantly from a good index, and support tools like `makeindex` or `texindex` can handle much of the drudge work of sorting and formatting the index entries.

LaTeX-mode provides two functions to support generation of index entries in the LaTeX file.

LaTeX-add-word-to-index on C-c . wraps an index entry around the word at the cursor position, either using a private indexing macro like `\X` described earlier on page 98, or by duplicating the word inside a standard LaTeX `\index{}` command. The choice between the two is determined by the Emacs variable *LaTeX-index-macro*; a non-nil value supplies the indexing macro name. With a numeric argument, multiple words can be selected for indexing.

LaTeX-index on C-c x generates

```
\index{□

}
```

so you can type an explicit index entry that will not appear at that point in the text.

For use in the X Window System on workstations and personal computers acting as X clients, `latex.el` provides additional functions, x-LaTeX-index-start and x-LaTeX-index-end that are bound to mouse actions; they cannot usefully be invoked from the keyboard.

To index a phrase in the document like *gnats and gnus*, simply click the right mouse button at the start of the first word, and then holding the button down, sweep the mouse across the entire phrase, lifting the button anywhere in the last word. Voilà: the buffer now contains `\X{gnats and gnus}`!

I find it useful to sit down with a typeset copy of a document, and use a colored highlighter pen to mark phrases to be indexed. With the marked-up copy, it is then straightforward to use the mouse to add the marked phrases to the index. For documents like this one, with lots of technical terms to be indexed, it pays to think about matters before you start writing. Define suitable macros that will typeset those terms in a particular font, and also automatically create an index entry for them. For example, I typeset and index an Emacs function using a private `\FUNCTION{}` macro.

The Emacs variable *LaTeX-index-start-with-newline* can be set to a non-nil value if you prefer to have the inserted index entry start a new line. The variable *LaTeX-index-end-with-newline* can be set to a non-nil value to have a newline generated after the inserted index entry. The default for both is nil, so that no additional lines are generated.

6.15 Cross-references

LaTeX's `\label`, `\pageref`, and `\ref` commands provide a convenient way to generate cross-references to equations, figures, pages, sections, and tables. In LaTeX-mode, they are readily available on the keys C-c l (that's the letter 'l'), C-c p, and C-c r.

Since definitions with `\label` tend to be separated from their references, you may often find yourself unable to remember the exact name of the label you want. If you type the label name incorrectly, you won't discover the error until you have run LaTeX twice. To avoid this delay, two functions provide additional support:

- show-LaTeX-labels finds all labels in the buffer and displays them with their line numbers; I ran that function while I was writing this article, and got the output

  ```
  LaTeX labels in buffer ltxmode.ltx:
  "X-command" [line 1220]
  "customizing-lists" [line 2358]
  "fig-bindings-1" [line 856]
  "fig-bindings-2" [line 928]
  "fig-latex-index" [line 789]
  "fig-latex-mode-1" [line 659]
  "fig-latex-mode-2" [line 744]
  "fig-letter" [line 2561]
  "fig-startup-code" [line 1006]
  "gnu-emacs-size" [line 376]
  "indexing" [line 2048]
  "latex-support" [line 946]
  "mode-hook" [line 1310]
  "word-abbreviations" [line 1931]
  ```

  showing the labels in alphabetical order. With a numeric argument, the labels are shown instead in line number order.
- check-LaTeX-labels scans the entire buffer, looking for labels that are referenced, but undefined. At each such label found, a recursive edit is entered to allow you to supply the missing label; when you exit the recursive edit, the search for undefined labels continues from where it left off.

These functions locate label references by searching for the string defined by the variable *LaTeX-label-reference*; it contains a regular expression that normally matches both `\pageref` and `\ref`. If you have defined private macros that also receive labels as arguments, you may wish to extend it. For example, this document has `\FIGREF` and `\PAGEREF` macros that simplify cross-referencing.

Similarly, these functions locate label definitions by searching for the string defined by the variable *LaTeX-label-definition*, another regular expression that normally matches `\label`. You may wish to customize it if private macros also generate labels.

Multi-file documents pose a challenge: labels defined in one file may be referenced in another file, and it would be helpful to be able to show a list of all currently-defined labels when one is needed for a cross-reference.

It is not feasible for the code in `latex.el` to attempt to figure out what files are used for a particular document, nor is it acceptable to ask the user to provide those file names. Yet we somehow need to scan those files to find all of the labels.

The solution is that LaTeX can do part of this job for us: LaTeX processing of the master file produces in the corresponding `.aux` file all defined labels embedded in `\newlabel` commands. The function update-LaTeX-labels does the remainder of the job: it prompts for the name of the master `.aux` file, and then scans it to extract a list of labels from the `\newlabel` commands. This list is then *appended* to the current contents of the variable *LaTeX-aux-file-label-tags*. Augmentation, rather than replacement, permits selective construction of list subsets by multiple invocations of update-LaTeX-labels on different files. check-LaTeX-labels includes the values from *LaTeX-aux-file-label-tags* in its list of defined labels.

For convenience, whenever LaTeX-mode is run (normally when a file is first visited), it will search the buffer for the first TeX macro which is preceded only by optional whitespace on its line. If that macro is `\documentstyle`, then the file must be the master file for the document, so *LaTeX-aux-file-label-tags* is set to nil, and update-LaTeX-labels is automatically called on the corresponding `.aux` file. This means that manual invocation of update-LaTeX-labels is needed only when you first visit a sub-file without first visiting the master file.

In recent versions of `latex.el`, LaTeX-pageref and LaTeX-ref have been augmented by LaTeX-pageref-with-completion and LaTeX-ref-with-completion. The new functions are bound to C-c p and C-c r, replacing the bindings to the older functions. The new ones use an internal function to dynamically scan the buffer for `\label{}` definitions, add the contents of *LaTeX-aux-file-label-tags* to it, and construct a list that can be used for command completion. They then prompt for a label, and if you type a query, they will display the current list of labels. You can use completion to select any one of them, or you can enter an arbitrary label that is not in the completion list.

Here is what the completion list looks like for this document:

```
Possible completions are:
X-command             customizing-lists
fig-bindings-1        fig-bindings-2
fig-latex-index       fig-latex-mode-1
fig-latex-mode-2      fig-letter
fig-startup-code      gnu-emacs-size
indexing              latex-support
mode-hook             word-abbreviations
```

Because the dynamic label scan that happens each time these functions are executed may be slow in large documents, the old functions remain available for rebinding to keys.

The Emacs variable *LaTeX-label-start-with-newline* can be set to a non-nil value if you prefer to have the inserted label entry start a new line. The variable *LaTeX-label-end-with-newline* can be set to a non-nil value to have a newline generated after the inserted label entry. The default for both is nil, so that no additional lines are generated.

6.16 Miscellaneous functions

LaTeX-footnote on C-c f generates a `\footnote{}`. Not only does this save typing, but it also prevents my fingers from typing `\bootnote{}`, which looks reasonable enough, but makes TeX unhappy.

LaTeX-news on C-c C-n views the `latex.el` file to show the revision history recorded in it.

LaTeX-tab on C-c Tab indents to the current `\begin{...}` ... `\end{...}` nesting level.

LaTeX-protect on C-c C-p inserts a `\protect` macro at the current point. It is needed whenever fragile commands (e.g. any LaTeX command that has a star form [9, p. 27]) are present in a moving argument [9, p. 151]. In particular, all macros in `\index{}` entries should be preceded by `\protect`. Otherwise, you can get long macro expansions that cause buffer-overflow problems for indexing programs, and make the index file hard to read. Also, macro expansions can introduce special characters which cause problems for indexing programs like `makeindex`.

LaTeX-set-indentation on C-c , sets the current indentation column to the value of its argument, or to the cursor column if there is no argument. It is only rarely necessary to use this function, because the indentation is reset by several other functions.

LaTeX-verb on C-c v generates a `\verb|...|` command with the cursor after the first vertical bar. With an argument, it creates the `\verb*|...|` form, which makes spaces visible. The delimiter character can be changed from a vertical bar by reassigning the Emacs variable *LaTeX-verb-delimiter*. For example, to set it to a plus sign instead of a vertical bar:

```
(setq LaTeX-verb-delimiter ?\+)
```

The funny `?\` prefix is one of Lisp's character quotes. As shown on page 98, this assignment should normally be placed inside a mode hook. It can also be executed dynamically inside the Emacs minibuffer.

There are several other functions in `latex.el` that I shall not document here; they are intended for internal use only, and are subject to change without warning.

6.17 Miscellaneous variables

There are a number of variables that we have not yet described. You should not have to modify any of them, but you occasionally might want to know what they are.

LaTeX-current-indentation tracks the current environment indentation as the number of spaces from the left margin. It is updated dynamically by many functions.

LaTeX-mode-abbrev-table holds the standard word abbreviations, which are further described on page 103.

LaTeX-mode-map holds the map that associates keys with functions.

LaTeX-mode-version is a string containing the version number and date of the last revision to the code. You should cite its value when reporting bugs; LaTeX-gripe puts it in the e-mail subject line.

LaTeX-option-list holds the options collected by make-LaTeX-document and make-SLiTeX-document.

LaTeX-standard-environments contains the list of environment names used by LaTeX-begin-end-block.

LaTeX-standard-fonts holds a list of font names and sizes.

LaTeX-standard-italic-fonts is a list of the LATEX control sequences for fonts that require italic corrections.

LaTeX-standard-list-environments is a list of LATEX environments which can have `\item` entries.

LaTeX-standard-options is a list of options that can go in the square brackets of the `\documentstyle` command. It is used by make-LaTeX-document and make-SLiTeX-document.

LaTeX-standard-styles is a list of styles that can go in the braces of the `\documentstyle` command. It is used by make-LaTeX-document.

LaTeX-standard-unindented-environments is a list of environments that must not be indented.

6.18 Customizing lists

It is undesirable for users to have to duplicate the long initializations of standard fonts, macros, environments, and so on, when all they want to do is add a few more items to the lists.

Therefore, for each variable *LaTeX-standard-xxx*, there is a corresponding variable *LaTeX-extra-xxx* that is intended for user customization. The extra values are appended to the standard ones to make new lists that are used for command completion. Here are the names of these customization variables: *LaTeX-extra-dotted-abbreviations*, *LaTeX-extra-environments*, *LaTeX-extra-fonts*, *LaTeX-extra-italic-fonts*, *LaTeX-extra-list-environments*, *LaTeX-extra-macros*, *LaTeX-extra-options*, *LaTeX-extra-styles*, and *LaTeX-extra-unindented-environments*.

Suppose for example that you want to add new environments named `conjecture`, `postulate`, and `wildguess` to the standard list used by LaTeX-begin-end-block. In the *LaTeX-mode-hook* in your `.emacs` file, add this assignment:

```
(setq LaTeX-extra-environments
      '(
        ("conjecture")
        ("postulate")
        ("wildguess")
        )
      )
```

When you subsequently type C-c b, these three new names will appear in the environment name completion list. Such lists are automatically alphabetized during the completion process, so the names can be given in any order in the assignment.

There is no provision for deleting names from the standard lists; after all, they are *standard*! Also, remember that the new additions are made known only to Emacs; you still have to teach LATEX how to typeset them.

You may have wondered in the example above why I wrote `("conjecture")` instead of just `"conjecture"` in the assignment of *LaTeX-extra-environments*. The extra parentheses make a list out of each element in the variable. In `latex.el`, all variables that hold multiple strings store them as *lists of lists* of strings, rather than as simpler lists of strings. This choice was intentional, because it facilitates adding additional related strings in the form of Lisp *association lists*.

The variables *LaTeX-extra-environments*, *LaTeX-extra-macros*, *LaTeX-standard-environments*, and *LaTeX-standard-macros* already use association lists to hold trailing braces and brackets, descriptive comments, and positioning functions.

7 SLITEX support

SLITEX works much like LATEX, so little additional support is required. All of the commands in the preceding sections can be used, and the editing mode is still called LaTeX-mode.

When you select a new SLITEX file to edit, you can create an initial template with make-SLiTeX-document on C-c s. It prompts only for document style options, since the major style is fixed at `slides`, and produces something like this:

```
% -*-latex-*-
% Document name: /u/beebe/tex/tugboat/foo.stx
% Creator: Nelson Beebe [beebe@math.utah.edu]
% Creation Date: Mon Jul 29 08:15:15 1991
\documentstyle[]{slides}
\begin{document}
  \onlyslides{1-9999}
  \onlynotes{1-9999}
% \colors{} % {red,green,blue,etc.} colors
% \colorslides{} % input file

  \blackandwhite{□} % input file

\end{document}
```

The cursor is left positioned in the brace pair of the `\blackandwhite{}` macro, where you can type the name of another file that you can create in LaTeX-mode, using `slide` environments generated by C-c b sli␣ RETurn.

It is convenient to number each slide with a comment, so that you have the numbers available if you need them for the `\onlynotes` and `\onlyslides` commands. The function renumber-slides on C-c 0 modifies relevant commands in the buffer to read something like this:

```
  \begin{slide}{} % slide number 1
```

```
...
  \begin{slide}{} % slide number 2
...
  \begin{overlay}{red} % overlay number 2-a
...
  \begin{overlay}{green} % overlay number 2-b
...
  \begin{note} % note number 2-1
...
  \begin{slide}{} % slide number 3
...
  \begin{note} % note number 3-1
...
  \begin{note} % note number 3-2
```

Any existing comments on the commands are discarded. Notice the special handling accorded the SLiTeX `note` and `overlay` environments; they receive a major number equal to that of the closest preceding slide, followed by a hyphen and a minor number or letter [9, pp. 136–137].

The Emacs variable *SLiTeX-begin-slide* defines the regular expression used to find the start of the `note`, `overlay`, and `slide` environments.

8 Letter support

LaTeX [9, pp. 66ff] provides a `letter` document style, but there is a fair amount of constant 'boilerplate' that has to be typed each time you use it. This suggests that the best approach is to have private template files that already have a letter outline in them, and then arrange to start each letter with that file already inserted into the buffer. Emacs' programmability makes this straightforward.

The startup file code in Figure 6 on page 96 loads a small Lisp file, `letter.el`. It defines a letter-mode function, and associates it with files with extension `.ltr`. If I then visit a new file `rb-jones.ltr`, I immediately get the buffer shown in Figure 7, with the cursor positioned in the empty brace pair in `\begin{letter}{}`, ready for entry of the address. The query in `\opening{Dear ?}` is a reminder to enter something there, and then the body of the letter can be typed in following the `\opening` line.

My personal style option `nhfb-letter` provides for a University and Department letterhead on the first page, and defines a few private macros that I often need in my correspondence.

Once the letter file has been saved, I suspend Emacs, or switch to another workstation window, and then process the letter by typing

```
make LET=rb-jones
```

```
% -*-LaTeX-*-
% Document name: /u/beebe/rb-jones.ltr
% Creator: Nelson Beebe [beebe@math.utah.edu]
% Creation Date: Mon Jul 29 20:06:18 1991

%----------------------------------------------
% EVERYTHING TO THE RIGHT OF A  %  IS A REMARK
% TO YOU AND IS IGNORED BY LaTeX.
%
% WARNING!  DO NOT TYPE ANY OF THE FOLLOWING
% 10 CHARACTERS AS NORMAL TEXT CHARACTERS:
%       &   $   #   %   _    {   }   ˆ   ˜   \
%
% The following seven are printed by typing a
% backslash in front of them:
%       $  &  #  %  _  {  and  }.
%----------------------------------------------

\documentstyle[nhfb-letter]{letter}
\begin{document}
\begin{letter}{□}

\opening{Dear ?}

\closing{Sincerely,}
\ps{{$\cal NHFB$}/\LaTeX}
\end{letter}
\end{document}
```

Figure 7: Letter template.

The UNIX `make` utility is a wonderful tool; here it directs the execution of commands to expand tabs to blanks, changes the file protection to remove access to all but its owner, runs LaTeX, and then runs a PostScript DVI driver, sending the output to a nearby printer. Then it sends the output PostScript through a shell filter that modifies the `showpage` command to print the string *File Copy* in outline letters diagonally across each page, and queues that to the printer as well. On those occasions where I require multiple copies of the original letter, I just type something like

```
make LET=rb-jones C=3
```

This gives me one file copy, and three originals for signing and mailing.

With this support, preparation of typeset letters is just as easy as typewritten ones, and the result looks much better. And I can use mathematics and a variety of fonts too!

9 Bug reporting

LaTeX-gripe on C-c g makes it easy to complain about LaTeX-mode; it puts you in an e-mail buffer with a mail header something like this:

```
To: beebe@math.utah.edu
Subject: LaTeX-mode gripe report {beta test
         0.23 [08-May-1991]}
```

The standardized subject field makes it easy for the author to identify such messages in his voluminous e-mail correspondence, and also automatically identifies the code version.

If you don't have an electronic mail connection yet, you probably soon will. For now, just send me your comments in postal mail. Machine-readable submissions can be sent on IBM PC or Apple Macintosh floppy disks.

10 Conclusion

LaTeX-mode editing support in GNU Emacs provides a very powerful, and pleasant, way to prepare LATEX and SLITEX input.

The original TOPS-20 version written in TECO was begun in October 1984, and used extensively at Utah until our DEC-20/60 retired on October 31, 1990. It consists of about 1000 lines of code defining 24 functions.

The power of Lisp over TECO made it possible to be much more ambitious in the functionality of the GNU Emacs version, and I consequently suspended further development of the TECO code in January 1988 when I began the Lisp coding. The current Lisp version is nearly 4 times the size of the TECO one, both in lines of code, and in editing functions. Despite the increase in size, it is *much* easier to maintain.

The GNU Emacs implementation was released for beta testing in March, 1988. About 70 sites have participated in the beta test of `latex.el`. I thank these many people for their contributions of comments, and sometimes, even code. Their names are recorded in the revision history in `latex.el`.

Is it bug free? No computer program for a realistic problem ever is; there are simply too many details for humans ever to get completely right. The `latex.el` revision history shows mostly additions of new functionality, and code changes for better performance. The virtue of the Emacs editing environment is that new functions can be written, debugged, and tested in the editor itself, providing immediate feedback and confidence in the correctness of operation.

Much of the code is in small functions that are independent of the others, following the Lisp programming tradition. After removal of comments and empty lines, and ignoring the initializations of large tables, the average is about 22 lines of code and documentation per function. This is close to the 20-line average for all of GNU Emacs, computed from the statistics given earlier on page 92. Documentation contributions are included here because Emacs functions, and some variables, carry their own documentation with them, even after compilation. That documentation is readily available for viewing in the editor.

I believe that this code could serve as a model for other Emacs-like editors that have an extension language in which new functions can be written. I had planned to write an EEL version for the `epsilon` editor on PC DOS, but alas, have not found the time to do so. This is not a small job, but it should be straightforward for someone familiar with both Lisp and EEL. The `latex.el` file is over nearly 3900 lines long, with about 142KB of text; Emacs compiles it into about 92KB of byte codes. The latter figure should give some idea of the memory requirements for implementations in other editors. I hope some reader will prepare such a translation and make it freely available, as `latex.el` is. A large portion of the TEX community uses personal computers that are inadequate for GNU Emacs, but have smaller Emacs-like editors available, and such editing support would be a great boon for them.

Some other work in the TEX community for editing support has been reported in the TEX Users Group TEXniques series [11, 12, 16]. I believe that `latex.el` is much more powerful, however.

Those who use UNIX workstations or VAX VMS systems where GNU Emacs is already available should eventually find `latex.el` in the `emacs/lisp` directory, and until then, can obtain it directly from the author, provided they have Internet electronic mail or `ftp` access.

If you have a UNIX or VAX VMS system, but do not yet have GNU Emacs, you can get Emacs from any site that already has it, or better, you can get the latest version directly from the Free Software Foundation. The latter is accessible either via Internet anonymous `ftp` to the machine `prep.ai.mit.edu`, or through postal mail to

Free Software Foundation, Inc.
675 Massachusetts Ave
Cambridge, MA 02139
USA

The postal mail approach carries some shipping and documentation charges, but the software itself is free.

References

[1] James Alexander. `Tib`: a reference setting package for TEX. *TUGBoat*, 7(3):138, October 1986.

[2] James Alexander. `Tib`: a reference setting package, update. *TUGBoat*, 8(2):102, July 1987.

[3] R. L. Aurbach. User's guide to the IdxTEX program. *TEXniques, Publications for the TEX community*, (3):i, 1–14, 1987.

[4] David R. Barstow, Howard E. Shrobe, and Erik Sandewall. *Interactive Programming Environments*. McGraw-Hill, 1984.

[5] J. L. Bentley and B. W. Kernighan. Tools for printing indexes. *Electronic Publishing—Origination, Dissemination, and Design*, 1(1):3–18, April 1988.

[6] Pehong Chen and Michael A. Harrison. Automating index preparation. Technical Report 87/347, Computer Science Division, University of California, Berkeley, CA, USA, March 1987. This is an expanded version of [7].

[7] Pehong Chen and Michael A. Harrison. Index preparation and processing. *Software—Practice and Experience*, 19(9):897–915, September 1988. The LaTeX text of this paper is included in the `makeindex` software distribution.

[8] Donald E. Knuth. *The TeXbook*. Addison-Wesley, 1984.

[9] Leslie Lamport. *LaTeX—A Document Preparation System—User's Guide and Reference Manual.* Addison-Wesley, 1985.

[10] Leslie Lamport. `MakeIndex`*: An Index Processor For LaTeX*, 17 February 1987. The LaTeX text of this paper is included in the `makeindex` software distribution.

[11] Kent McPherson. VAX Language-Sensitive Editor (LSEDIT) Quick Reference Guide. *TeXniques, Publications for the TeX community*, (1):ii, 1–9, 1988.

[12] Paul M. Muller. FASTTeX: A PC text editor and front-end for TeX. *TeXniques, Publications for the TeX community*, (7):235–254, 1988.

[13] Rainer Schöpf. A new implementation of the LaTeX `verbatim` and `verbatim*` environments. *TUGBoat*, 11(2):284–296, June 1990.

[14] Margaret Shertzer. *The Elements of Grammar*. Collier Books, Macmillan Publishing Company, 1986.

[15] Harriett Beecher Stowe. *Uncle Tom's Cabin, or Life Among the Lowly*. Oxford University Press, 1965. First published in 1852.

[16] Stephan von Bechtolsheim. Using the Emacs editor to safely edit TeX sources. *TeXniques, Publications for the TeX community*, (7):195–202, 1988.

Index

Prolegomena toward a font selection scheme

Victor Eijkhout

Department of Computer Science University of Tennessee
104 Ayres Hall
Knoxville, Tennessee 37996, USA

eijkhout@cs.utk.edu

1 Introduction

Most users of plain TEX do not get very sophisticated in their use of fonts. Often they resort to declaring all used fonts explicitly with `\font`. There are some obvious disadvantages to that: it is not possible to switch a whole document in a simple way to a different typeface, or to a different size. As a result, I've seen such phenomena as an article with an abstract in 8 or 9 point, but where the formulas were still in 10 point, or pages of 'magnified' type where the lines were cramped, because the `\baselineskip` was not increased with the type size.

The need for a good font selection scheme is thus quite obvious, but the implementation of one is not.

2 Concepts

2.1 Font parameters

One point that most people agree on is that fonts are characterized by a number of parameters, and that it should be possible in an easy way to select the font that arises from changing just one of those parameters. For instance, switch to italics for emphasis, switch to a smaller size for a footnote, switch to the Greek alphabet to give an erudite – and untranslated, of course – quote from Homer.

However, the exact number of parameters is a point of contention. Appendix E of TEX Users Group, [2], gives macros that are based on size and style (`\rm`, `\it`, et cetera). An obvious third parameter – which Knuth, working exclusively in Computer Modern did not need – is the typeface. In [3, 4] Frank Mittelbach and Rainer Schöpf use an extra parameter, splitting style into 'shape' and 'series'. The series parameter combines the typographical qualities weight and width.

Furthermore, Karl Berry in [1] splits the typeface parameter into 'foundry' and 'typeface family'[1], and Yannis Haralambous suggested that the parameter 'alphabet' is also needed.

An international standard for font properties even exists: in section 8.6 of [5] 23 font properties are given. However, not all of these can be independently varied. For instance, the design copyright and the list of excluded characters are merely informational. The parameters suitable for inclusion in a font selection scheme are here: typeface name, posture (upright, oblique, italic, ...), weight (light, medium, bold, ...), proportionate width (condensed, medium, expanded, ...), structure (solid, outline, ...), and design size. Notably missing is the alphabet parameter.

2.2 More parameters, less parameters; do we have an argument?

To a certain degree, the above font classifications clash less than they seem to. Most parameters are orthogonal: if parameter 1 has p_1 values and parameter 2 has p_2 values, then all $p_1 \times p_2$ values make sense. But in that case we can introduce a new parameter 3 which has that many values. Thus the above schemes differ mostly in how far they split up the variety of fonts, not along what lines.

The number of parameters is to some extent a matter of taste, but a few common sense observations can be made.

- It is possible to live completely without any parameters by using only the control sequences that have been defined with `\font` declarations. This is very inconvenient.
- Certain DTP packages make it easy for you to (produce horrible typography: you simply) switch to 'outline' or 'shadow' (or both) for whatever font you are currently using. This is no reason to introduce independent parameters 'outline' and 'shadow' for every conceivable mutilation of typefaces.
- Some parameters are not likely to get varied: it is hard to conceive how the same font will be used from two different foundries in the same text, other than in texts that write specifically about typeface comparisons.
- Readers may have wondered why the distinction between serif and sans serif has not come up yet. The reason is that this is not an orthogonal category. Most typefaces exist only in seriffed or serifless

[1] It should be noted that this is not a proposal for of a font selection scheme, but for systematic filenames. The number of user parameters can either be more or less than the properties encoded in the file names.

design, but not both. For the occasional exception (Computer Modern, for instance) it is easiest to pretend that the seriffed and serifless subfamilies are simply two typeface families.

3 Implementation

The previous section described some of the issues involved in providing the user with suitable parameters for handling fonts. We must now look at the other side of the question: the parameters have to be translated into the name of a `tfm` file.

Any implementation of a font selection scheme has to pay attention to a couple of points.

- Certain combinations of parameters may not correspond to an existing font. A substitution may have to be performed, and in any case the user has to be told.
- For large sizes there must be a mechanism that makes a reasonable guess as to the file name.
- The memory usage of the font selection scheme should be as low as possible.

The last point can be elaborated a bit: memory usage should increase in a controlled manner if more parameters or more typefaces are added. Adding parameters may seem a bit strange to do, but it makes sense for such parameters as ‘structure’: suppose all fonts so far have been solid, and suddenly a printer is bought that can make an outline from any given font. Is the amount of memory for the font selection scheme then increased by a small quantity, or is it doubled?

Specifying for every combination of all parameters what file name results (as is done in the font selection scheme in [3]; in order “to implement the four dimensional grid of fonts” it keeps “for every combination of font family/series/shape” a list of size/external name pairs) is in one sense the optimal solution: any system (or absence thereof) of file names can be accommodated. However, the space needed is proportional to the product of all possible values of all parameters.

On the other hand, if file names are systematic (as in the naming scheme in [1], and as in the Computer Modern typefaces and all Bitstream and PostScript typefaces as far as I’ve seen), then considerable savings are possible: it becomes possible to translate a few parameters jointly into a fragment of the file name, and concatenate such fragments into the full file name. This reduces storage to a higher power root of the original amount[2].

This approach of componentwise translation offers an additional advantage for the ‘size’ parameter: for PostScript fonts the size can be given procedurally, for instance specifying

```
size: at #1pt
```

thus taking a fixed amount of storage for an infinite number of values.

An entirely modular translation is probably not possible. For instance, the translation of shape and series into a file name fragment is most likely dependent on the typeface, although it may be the same for all typefaces from a certain foundry.

Another impediment to modular translation is the handling of special cases. For instance, for certain typefaces certain styles may not be available, or styles may not be available in all sizes (probably the smaller sizes). However, such exceptions can most likely still be treated in a procedural manner.

References

[1] Karl Berry. Filenames for fonts. *TUGboat*, 11:517–520, 1991.

[2] Donald E. Knuth. *The TEXbook*. Addison Wesley, 1984. reprinted with corrections 1989.

[3] Frank Mittelbach and Rainer Schöpf. A new font selection scheme for TEX macro packages. *TUGboat*, 10:222–238, 1989.

[4] Frank Mittelbach and Rainer Schöpf. The new font family selection – user interface to standard LATEX. *TUGboat*, 11:297–305, 1990.

[5] ISO standard. Information technology – Font information interchange – Part 1: Architecture; international standard ISO/IEC 9541-1, 1991.

[2] For instance, translating three parameters independently makes storage proportional to the sum of the values of the three instead of the product. Thus storage is proportional to the cube root of what it would be in a fully explicit scheme.

From observation to publication

Theo A. Jurriens

Kapteyn Astronomical Institute P.O. Box 800
9700 AV Groningen
The Netherlands

`taj@rugr86.rug.nl`

Abstract

This article describes the use of TeX in publishing observations of variable stars observed by Dutch amateur-astronomers. The observations are published in the journal "Variabilia" and in the so-called Reports. In the latter the observations, collected in several years, are published and submitted to the professional astronomer. It includes tables and light-curves: plot of the changing magnitude of the star versus time. In creating the light-curves: PiCTeX is used. In preparing the files for PiCTeX simple TeX-coding is used for manipulating the data.

1 The data

Each observation is characterized by the observed star, a date, the brightness of the star and a code representing the observer. To avoid calender problems the so-called Julian Date is used. For example the Julian Date of 1961, september 9, my date of birth, is equal to 2437552 .

Each observation is TeX- coded as followed:

`\obs #1.#2.#3.#4.#5.`

`#1` is the integer of the Julian Date, `#2` it's fraction. `#3` is the integer of the magnitude, `#4` it's fraction and `#5` is the observer-code. The latter is not used for plotting. TeX can calculate with integers so the observing date and magnitude, the apparent brightness of a star, are transformed to integers. In case of date it is simply the integer of the Julian date. The magnitude is expressed in units of 0.1 so the magnitude is $10 \times \#3 + \#4$.

For each star we have a number of observations. The file of observations processed by `plain`- TeX produces a file suitable for PiCTeX. This `.plt` file contains all the necessary information for a plot: the axis are proper scaled and labeled and contains information about the star. See the example at page 119 Figure 1 shows an typical file ready for TeX.

The macros used to create the `.plt` file are given below. Part II neglect the fraction of the Julian Date and calculates the magnitude, in units of 0.1. Both are written to a tempory file with the extension `.obs`. In part II also the minimum and maximum values along the axis are estimated. These values are used in Part III to calculate the proper sizes of the graph. In Part III the final `.plt` file is created and can be processed by PiCTeX.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  PART I plot.tex     %%
%%%%%%%%%%%%%%%%%%%%%%%%%%
\newcount\tempx    \newcount\tempy
\newcount\tempz    \newcount\maxx
\newcount\minx     \newcount\miny
\newcount\maxy     \newcount\numobs
\newwrite\plotfile \newwrite\obsfile
\def\head#1{
\global\def\plotname{\sternum\ #1}
\initplot}
\def\endhead{\immediate\closeout\obsfile
\startplot}
\def\harvard#1{\global\def\sternum{#1}}
\def\type#1{\global\def\typename{#1}}
\def\initplot{
\immediate\openout\obsfile=\jobname.obs
\global\minx=99999
\global\maxx=0
\global\miny=99999
\global\maxy=0
\global\numobs=0}
\def\pplot#1#2{\tempx=#1
\tempy=\maxy
\tempz=#2
\advance\tempy by-\tempz
\advance\tempx by-\minx
\immediate\write\plotfile{\noexpand
\put {$\noexpand\bullet$} at
{\the\tempx} {\the\tempy}}}
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  PART II             %%
%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\obs #1.#2.#3.#4.#5.{
\message{#5}
\global\advance\numobs by 1
\tempx=#1
% estimate min max x
\ifnum\minx>\tempx \global\minx=\tempx \fi
\ifnum\maxx<\tempx \global\maxx=\tempx \fi
\tempy=#3
\tempz=#4
% calculating magnitude
\multiply\tempy by 10
\advance\tempy by\tempz
% estimate min max y
\ifnum\miny>\tempy \global\miny=\tempy \fi
\ifnum\maxy<\tempy \global\maxy=\tempy \fi
\immediate\write\obsfile{\noexpand
\pplot{\the\tempx}{\the\tempy}}}
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  PART III            %%
%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\startplot{
% create nice numbers along axis
\global\divide\minx by 10
\global\multiply\minx by 10
\global\divide\maxx by 10
\global\multiply\maxx by 10
\global\advance\maxx by 10
\global\divide\miny by 10
\global\multiply\miny by 10
\global\divide\maxy by 10
\global\multiply\maxy by 10
\global\advance\maxy by 10
\global\advance\maxx by-\minx
\ifnum\numobs>15
% writing pictex commands
 \def\name{\sternum.plt}
 \immediate\openout\plotfile=\name
 \message{\name}
 \immediate\write\plotfile{\noexpand
 \beginpicture}
 \immediate\write\plotfile{\noexpand
 \setcoordinatesystem units <0.35mm,1mm> }
 \tempz=\maxy
 \advance\tempz by-\miny
 \immediate\write\plotfile{\noexpand
 \setplotarea x from 0 to {\the\maxx},
 y from 0 to {\the\tempz}}
 \immediate\write\plotfile{\noexpand
 \plotheading{\plotname\ \typename}}
 \immediate\write\plotfile{\noexpand
 \axis bottom label {JD-{\the\minx}}}
 \immediate\write\plotfile{ticks
 numbered from 0 to {\the\maxx} by 40 / }
 \immediate\write\plotfile{\noexpand
 \axis left label {} }
 \immediate\write\plotfile{ticks
 from 0 to {\the\tempz} by 10 / }
 \tempx=\miny
 \advance\tempx by-10
 \tempz=\maxy
 \loop\ifnum\tempz>\tempx
        \tempy=\maxy
        \advance\tempy by-\tempz
        \immediate\write\plotfile{\noexpand
        \put {\the\tempz} at
        -15 {\the\tempy}}
         \advance\tempz by-10
 \repeat
% find out where to put text
% along y-axis.
 \tempz=\maxy
 \advance\tempz by-\miny
 \divide\tempz by2
 \tempx=\tempz
 \divide\tempx by10
 \tempy=\tempz
 \multiply\tempx by10
 \advance\tempy by-\tempx
 \ifnum\tempy=0
        \advance\tempz by5
 \fi
 \immediate\write\plotfile{\noexpand
 \put {Magn.} at -15 {\the\tempz}}
 \input \jobname.obs
 \immediate\write\plotfile{\noexpand\endpicture}
 \immediate\closeout\plotfile
\fi}
```

```
\input plot.tex
\harvard{000451}
\head{SS Cas}
\type{Mira  HIP}
\obs 47537.4.10.0.CMG.
\obs 47539.4.9.7.FJH.
\obs 47544.3.10.2.CMG.
\obs 47544.3.10.4.BMU.
\obs 47549.4.10.5.BMU.
\obs 47550.4.10.3.CMG.
\obs 47554.4.10.5.NWL.
\obs 47565.3.11.3.CMG.
\obs 47565.3.11.4.BMU.
\obs 47567.3.11.5.FJH.
\obs 47569.3.11.7.BMU.
\obs 47573.3.11.7.CMG.
\obs 47574.4.12.0.BMU.
\obs 47578.3.12.1.CMG.
\obs 47579.3.11.8.FJH.
\obs 47579.3.12.1.BMU.
\obs 47586.4.12.3.BMU.
\obs 47592.3.12.6.CMG.
\obs 47594.3.12.5.FJH.
\endhead
\bye
```

Fig. 1: *An input-file for TEX to create a file ready for PICTEX.*

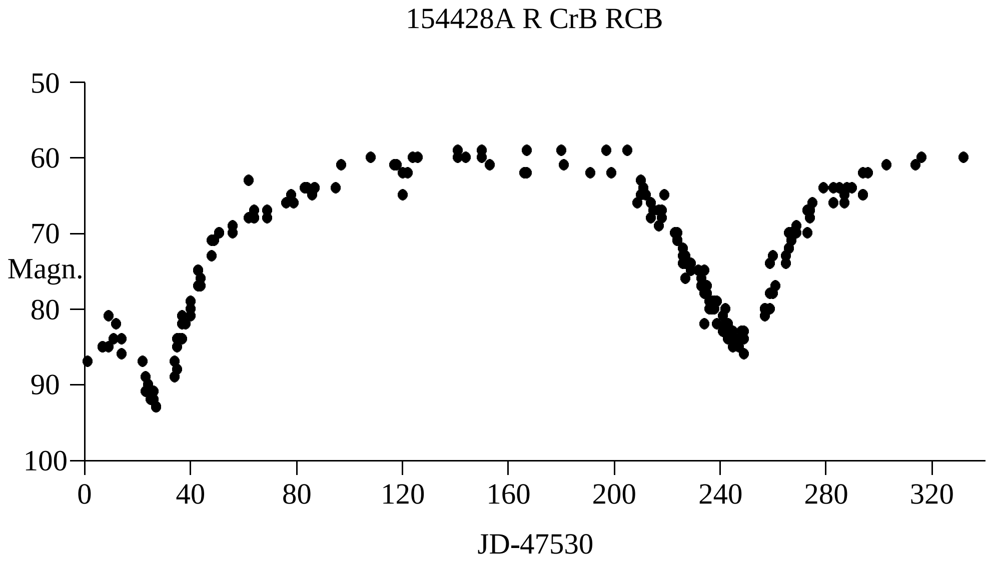


Fig. 2: *The light-curve of the star R CrB. The decrease in brightness can be seen clearly in this picture. The data are collected by Dutch amateur-astronomers.*

2 Macro's

In Part I the initialization is done: counters and files are defined. Information about the stars are stored in the tokens `\plotname`, `\sternum` and `\typename`. Also some initial values for the minimum and maximum values along the axis are set. The macro `\pplot` is used in Part II and executed in Part III. The macro writes the re-scaled X and Y values to the `.plt` file. Again the latter is used with PICTeX to obtain the desired result.

In Part II `\obs` is defined: the magnitude and Julian Date are converted to integers as described above and the maximum and minimum values are estimated according to the simple algorithme:
if$x_i <$ min then min $:= x_i$;
if$x_i >$ max then max $:= x_i$;
Also the number of observations is counted. This number is stored in `\numobs`.

In Part III the final plotting commands are written to the `.plt` file. This part is only executed if the number of observations is bigger than 15. The minima and maxima found in Part II are re-calcuted to have nice numbers along the axis. Commands for labels and numbering axis are also written into the file. After setting up the graph the data to be plotted are read from the `.obs` file. This file is created in Part II. Figure 3 shows such a `.plt` file.

```
\beginpicture
\setcoordinatesystem units <0.35mm,1mm>
\setplotarea x from 0 to {340},
y from 0 to {50}
\plotheading {154428A\ R CrB\ RCB}
\axis bottom label {JD-{47530}}
ticks numbered from 0 to {340} by 40 /
\axis left label {}
ticks from 0 to {50} by 10 /
\put {100} at -15 {0}
\put {90} at -15 {10}
\put {80} at -15 {20}
\put {70} at -15 {30}
\put {60} at -15 {40}
\put {50} at -15 {50}
\put {Magn.} at -15 {25}
\put {$\bullet $} at {1} {13}
\put {$\bullet $} at {7} {15}

⋮

\endpicture
```

Fig. 3: *The start of a typical plot-file as created by Part III.*

3 Publishing of the data

All the observations of one year are collected in one file with a structure as in Figure 1. This file contains 6000 to 10.000 lines depending on the weather conditions. The file is processed using `\plot.tex` and a number of `.plt` files are created. The same file is used to create a six-column tabular output of the observations, see the example at the next page. These macro's are available on request.

table page to be inserted by hand

FIFO and LIFO incognito

Kees van der Laan

February 1992

Abstract

FIFO, first-in-first-out, and LIFO, last-in-last-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.

1 Introduction

It started with the programming of the Tower of Hanoi in TEX, see van der Laan (1992). 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) paradigma will hopefully improve the situation.

2 FIFO

FIFO can be TEXed as template via[3]

```
\def\bfifo#1#2\efifo{\process{#1}
\ifx\empty#2\empty
\else\bfifo#2\efifo\fi
}%end \bfifo...\efifo
```

Printing of a tower can be done via

```
\def\process#1{\kern.2ex\hbox to3ex{%
\hss\vrule width#1ex height1ex\hss}}
\vbox{\offinterlineskip\bfifo12\efifo}
```

The `\bfifo...\efifo` macro is a basic one. It allows to procede along a list and to apply a (user) specified process to each list element. By this approach the programming of walking through a list is separated from the various processes to be applied to the elements. Fundamental![4]

The recursion will be terminated if #2 is empty.[5] One can circumvent the building up of `\fi`'s via[6]

```
\def\bfifo#1#2\efifo{\process{#1}
   \ifx\empty#2\empty
   \else\def\aux{\bfifo#2\efifo}
        \expandafter\aux\fi
}%end \bfifo...\efifo
```

or via

```
\def\bfifo#1#2\efifo{\process{#1}
   \ifx\empty#2\empty\let\aux=\relax
   \else\def\aux{\bfifo#2\efifo}
   \fi\aux
}%end \bfifo...\efifo
```

A more TEX-like implementation is

```
\def\bfifo#1{%
   \ifx\efifo#1\let\nxt=\relax%
   \else\def\nxt{\process{#1}\bfifo}%
   \fi\nxt}%end \bfifo
```

The advantage of the last implementation is that the input stream is processed one group or token at a time until `\efifo` is encountered. Moreover, it can handle the invoke `\bfifo\efifo`, the empty case. No auxiliary stacks are involved. This way of programming is unusual for those familiar with PASCAL-like programming.

[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] My first version had the two tokens after `\ifx` reversed — a cow flew by — and made me realize the non-commutativity of the arguments of TEX's conditionals. In math and in programming languages like PASCAL the equality relation is commutative! Note that at least one argument is needed in the above given implementation of FIFO.

[4] 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.

[5] Note that the second `\empty` is not always necessary. Knuth and Mackay (1987) demonstrate yet another variant of programming the test. The above given form is in agreement with Knuth's style as demonstrated in `\displaytest`, see the TEXbook, Appendix D-1, p.376.

[6] See Kabelschacht (1987).

2.1 Variable number of parameters

TEX macros can take at most 9 parameters. The above `\bfifo` macro can be seen as a macro which is relieved from that restriction. Every group or token in the input stream after `\bfifo` will become an argument to the macro. The first token or group is the first argument to the first invoke. This invoke ends with an invoke of itself using the next token or group from the input stream as argument. So the second token is argument of the second invoke. In general the n^{th} token or group is argument of the n^{th} invoke and so on until the `\efifo` token is reached, whereupon no invoke of `\bfifo` will occur.[7]

2.2 Length of string

An alternative to Knuth's macro, TB219, is obtained via an appropriate definition of `\process`.

```
\newcount\length
\def\process#1{\advance\length1}
```

Then `\bfifo aap\efifo` yields the length **3**.

2.3 Vertical printing

David Salomon treats the problem of vertical printing in his courseware. Via an appropriate definition of `\process` and a suitable invoke of `\bfifo..\efifo` it is easily obtained.

```
\def\process#1{\hbox{#1}}
xy\vbox{\bfifo abc\efifo}yx
```

yields xy$\begin{matrix}a\\b\\c\end{matrix}$yx.[8]

2.4 Delete last character of argument

Again an example due to David Salomon. It is related to the well-known `\gobble` macro to eat the *next* token (or group) from the input stream. One could define an appropriate `\process` but that will require double testing. Simpler is the following modification of the `\bfifo...\efifo` template.

```
\def\bgobblelast#1#2\egobblelast{
   \ifx\empty#2\empty\let\aux=\relax
   \else#1\def\aux{\bgobblelast#2%
    \egobblelast}\fi\aux%
}%end \bgobblelast...\egobblelast
```

Then `\bgobblelast aap\egobblelast` will yield `aa` .

2.5 To process words

In document preparation it is important to be able to handle quantities sequentially as elements of a list. What about handling a list of words? Amy Hendrickson in her courseware considers among others the problem of underlining words. This can be done by underlining every character, but that is slow. A faster solution can be obtained by first modifying the `\bfifo...\efifo` template into a version which picks up words, and to give `\process` the function to underline its parameter.[9]

```
\def\bfifow#1 #2\efifow{\process{#1}%
%Process words recursively;
%no addition of space here (part of
%\process if needed).
  \ifx\empty#2\empty\let\aux=\relax
  \else\def\aux{\bfifow#2\efifow}%
  \fi\aux}%end \bfifow...\efifow
```

The more TEX-like implementation, where the input stream is processed word wise, reads

```
\def\bfifow#1 {%
   \ifx\efifow#1\let\nxt\relax
   \else\def\nxt{\process{#1}\bfifow}
   \fi\nxt}%end \bfifow
```

2.5.1 Underlining words

In print it is unusual 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 `\process` definition underlines words picked up by `\bfifow...\efifow`.

```
\def\process#1{\vtop{\hbox{\strut#1}
              \hrule}\ }
\leavevmode\bfifow leentje leerde lotje
        lopen langs de lange lindenlaan
\efifow\unskip.
```

yields:[10]
leentje leerde lotje lopen langs de lange lindenlaan.

Note that underlining of complete sentences has to be considered separately if underlining punctuation marks is forbidden. (One possibility is to redefine `\process` such that the last symbol of its argument is inspected and appropriate action taken; another possibility is to use the general `\bfifo...\efifo` macro and suppress underlining for punctuation marks via appropriate programming of `\process`.)

3 Nested FIFO

One can nest the FIFO paradigma for example for processing lines word per word.[11] The template reads

```
\def\bfifol#1^^M#2\efifol{
  \bfifow#1\efifow
  \ifx\empty#2\empty\let\auxl=\relax
  \else\def\auxl{\bfifol#2\efifol}
  \fi\auxl}%end \bfifol...\efifol
```

with `\bfifow...\efifow` as defined above.

[7] Another way to circumvent the 9 parameters limitation is to associate names to the quatities to be used as parameters, let us say via def's, and use these quatities via their names in the macro. This is related to the so-called keyword parameter mechanism of command languages.

[8] Note the use of the `\hbox...` in process.

[9] Note that underlining inhibits hyphenation.

[10] `\unskip` is needed to undo the insertion of the last space.

[11] Or character per character, token per token, or group per group.

3.1 Natural data

Data for `\h(v)align` needs & and `\cr` marks. We can get plain TEX to insert an automatic `\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 invoke of `\halign`, one wants to get plain TEX to do the transformation

```
P*ON          P\cs*\csO\csN\rs
         ⇒
DEK*          D\csE\csK\cs*\lr
```

This can be done via adaptation of the above template along with an appropriate `\process` definition.

```
\let\ea=\expandafter
\newdimen\csize\csize=3ex
\def\rs{\cr}%generalization of row sep
\def\lr{\cr}%last row
\def\cs{&}%generalization of column sep
\catcode‘*=13 \def*{%crossed out cell
\vrule width0.6\csize height0.5\csize %
depth0pt}%simple BLACK variant
%more pleasing is the following
%poor man’s grey
\newbox\crs
\setbox\crs=\hbox to.6\csize{\leaders%
\hbox to.2ex{\hss\vrule height.5\csize
depth0pt\hss}\hfil}
\def*{\copy\crs}
%
\catcode‘\^^M=13 \let^^M=\relax
%Pick up and processing of lines
\def\bfifol#1^^M#2\efifol{%
\process{#1}%
\ifx\empty#2\empty\def\auxl{\lr}%
\else\def\auxl{\rs\bfifol#2\efifol}%
\fi\auxl}%ensure end conditional before
         %inserting tabular mark up
\def\process{\bfifo#1\efifo}%
%
%Pick up etc of chars per line
\def\bfifo#1#2\efifo{#1%#1 back
\ifx\empty#2\empty\let\aux=\relax%
\else\cs%insert \cs
\def\aux{\bfifo#2\efifo}\fi%
\aux}%
```

To demonstrate that it wor—-hey it works!—ks

```
%%%%%%% data provision %%%%%%%
\def\data{%
P*ON
DEK*
}%
%%%%%%% data transform %%%%%%%
\ea\def\ea\data\ea{\ea%
\bfifol\data\efifol}%define transform
%%%%%%%  application  %%%%%%%
$$\vbox{\halign{&\hbox to\csize{%
\vrule height.8\csize width0pt
depth.2\csize\hfil#\hfil}\cr\data}}$$%
```

will yield

```
P  ▥  O  N
D  E  K  ▥
```

As may be guessed from the layout the above came to mind when typesetting crosswords, while striving after the possibility to allow natural input, independent of `\halign` processing. Note that a weak form of the look ahead principle is implicitly applied as well.

4 LIFO

A modification of the `\bfifo...\efifo` 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 flight,’ without explicitly allocating auxiliary storage.[12]

```
\def\blifo#1#2\elifo{%
   \ifx\empty#2\empty\let\aux=\relax%
   \else\def\aux{\blifo#2\elifo}\fi%
   \aux\process{#1}%
}%end \blifo...\elifo
```

With the identity—`\def\process#1{#1}`—the template can be used for reversion. For example `\blifo aap\elifo` yields paa.

5 Further reading

Zalmstra and Rogers (1989), apply the FIFO technique to a list of figures — or floating bodies — in order to merge the list appropriately with the main vertical list in the output routine. This is beyond the scope of this paper.

6 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—TEX templates for FIFO and LIFO, emerged.

The templates can be used for processing lines, words or characters. Also processing of words or characters per line can be handled via nested usage of the FIFO principle.

TEX’s conditionals are non-commutative, while the similar mathematical and programming operations are.

From the application point of view the FIFO principle along with the look ahead mechanism is applied to molding natural data into representations required by subsequent TEX processing.

[12] 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 invoke 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.

References

[1] Hendrickson, A (priv. comm.)

[2] Kabelschacht, A (1987): `\expandafter` in conditionals; a generalization of plain's `\loop`. *TUGboat* 8, no. (2), 184–185.

[3] Knuth, D.E (1968): The Art of Computer Programming. 1. Fundamental Algorithms. Addison-Wesley.

[4] Knuth, D.E, P. Mackay (1987): Mixing right-to-left texts with left-to-right texts. *TUGboat* 7, no. (1), 14–25.

[5] Knuth, D.E (1984): The TEXbook. Addison-Wesley.

[6] Laan, C.G. van der (1992): Tower of Hanoi, revisited. *TUGboat* 13, no. (1), 91–94.

[7] Salomon, D (priv. comm.)

[8] Wirth, N (1976): Algorithms + Data Structures = Programs. Prentice-Hall.

[9] Zalmstra, J, D.F. Rogers (1989): A page make-up macro. *TUGboat* 10, no. (1), 73–81.

Tower of Hanoi, revisited

Kees van der Laan

December 1991

Abstract

Another version of programming 'The Tower of Hanoi' in TeX is provided.[1] No nodding knowledge of Lisp is required; just plain TeX. There is no restriction on the number of disks, apart from the installed limits of TeX. Generalized disks can be moved as well.

1 Introduction

At the Dedham TUG91 conference, I attended David Salomon's advanced TeX course. Instead of redoing his clear and ample exercises I decided to rework Leban(1985). The more so because elaborating a classic example might bring you to fundamental issues. In courses on programmming the Tower of Hanoi problem is used to illustrate paradigms. I was pleased to encounter some paradigms of TeX programming while revisiting the tower.

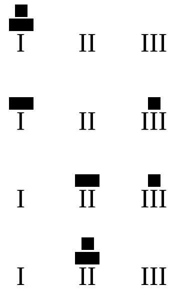
2 The Tower of Hanoi problem

A pyramid of disks—meaning a tower with implicit ordering of the disks—has to be moved under the restrictions that only one disk at a time can be moved and that each intermediate state consists of pyramids, obeying the original implicit ordering. In total three places for (intermediate) pyramids are allowed. For a pyramid of n disks, the solution needs $2^n - 1$ moves. For an introduction to the problem see the first paragraphs in Graham c.s.(1989).

3 Example

```
\input hanoi.tex \hanoi2 \bye
```

will yield the process of replacements for 2 disks from tower I to tower II



4 The file hanoi.tex

This file contains all the macros: the top `\hanoi`, the version with more parameters `\Hanoi`, the macro for the moves `\movedisk`, along with the auxiliary macro to prefix a string, `\logoffx...\logoffx...`, and the auxiliaries for printing `\showtowers`, and `\pt`.

As data structure a simplified version of Knuth's list, see TeXbook Appendix D.2, is used. A tower has the replacement text \\⟨item$_1$⟩ \\⟨item$_2$⟩ . . . \\⟨item$_n$⟩, where in this case each item is a control sequence. The separators, \\, '. . . are enormously useful, because we can define \\ to be any desired one-argument macro and then we can *execute* the list!'

```
%hanoi.tex version 18 dec 91
\newcount\n   %The number of disks
\newcount\brd %Breadth of towers
\newcount\hgt %Height of maximum tower
\newcount\dskhgt %Height of each disk
\let\ea=\expandafter %Shorthand
\let\ag=\aftergroup  %Shorthand
\def\preloop{%To create loopcnt, a
             %local loopcounter
             %(see also loopy.TeX).
   \bgroup \advance\count10 by 1
   \countdef\loopcnt=\count10
            %Symbolic name
   \loopcnt=1 %(default)
 }%end \preloop
\def\postloop{\loopcnt=0 %Restore
   \egroup}%end \postloop
%
%Hanoi macros, top level
\def\hanoi#1{%Argument can be digit(s)
             %or a counter (numeric)
 \n=#1 %Assign argument value to \n
 \def\II{}\def\III{}%Empty towers
 %Next is inspired by the TeXbook,
 %p374, 378
 %The initial tower for \I is created
 %The initial tower for \I is created
```

[1] For an earlier article on the issue, see Leban(1985).

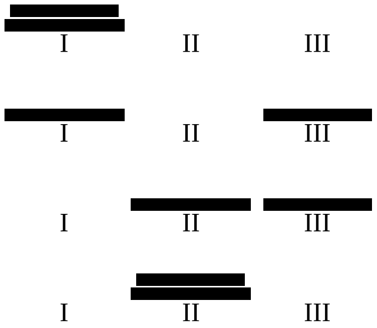
```
 %   \def\I{\\\i\\\ii\\\iii...\\\‘n’}
 %next to the defs for \i,\ii,...\‘n’.
 \preloop\ag\def\ag\I\ag{%
  \loop
    \ea\xdef\csname\romannumeral\loopcnt
            \endcsname{\the\loopcnt}
    \ag\\%separator
    \ea\ag\csname
         \romannumeral\loopcnt\endcsname
    \ifnum\loopcnt<\n
    \advance\loopcnt by 1
  \repeat   \ag}
 \postloop
 %For printing, values are needed for
 \brd=\n %Breadth of largest disk
 \advance\brd by 3 %Little room extra
 \dskhgt=1 %Height of disks
 \hgt=\n\multiply\hgt by2 %\hgt is height
       \advance\hgt by1 %of towers
 \showtowers %Print initial state
 \Hanoi\I\II\III\n
 }%end \hanoi
%
\def\Hanoi#1#2#3#4{%Moves from #1 to #2,
                  %with aid of tower #3.
 %The number of disks is #4, in a counter.
 \ifnum#4=1 %For Tower of 1 disk,
           %just move the disk
   \movedisk\from#1\to#2%
   \showtowers%Print towers after move
 \else%Problem of #4 disks is solved by
      %- problem of (#4-1) disks,
      %- a move, and
      %- a problem of (#4-1) disks.
   {\advance#4 by-1 \Hanoi#1#3#2#4}%
   \movedisk\from#1\to#2%
   \showtowers%Print towers after move
   {\advance#4 by-1 \Hanoi#3#2#1#4}%
 \fi}%end \Hanoi
%
%Moving of the disks, TeXbook, App. D.2
%Slightly adapted versions of \lop (
%called \movedisk with function that
%first element of #1 is prefixed to #2)
%and \lopoff modification
\def\movedisk\from#1\to#2{%Move disk from
%tower #1 to tower #2
  \ea\lopoffx#1\lopoffx#1#2}
\def\lopoffx\\#1#2\lopoffx#3#4{\ea\gdef%
  \ea#4\ea{\ea\\\ea#1#4}
  \gdef#3{#2}%restore stub}%end\lopoffx
 }%end \movedisk
%
%Printing tower status
\def\showtowers{%Display pyramids
 \par\quad\hbox{\pt\I\ \pt\II\ \pt\III
              }\par
 }%end \showtowers
%
%Auxiliaries
\def\gobble#1{}%To eat character
%
\def\\#1{\hbox to\brd ex{\hss%
  \vrule width#1ex height\dskhgt ex%
                      \hss}%
 }%end \\
%
\def\pt#1{%Print Tower.
%#1 is \I, \II, or \III
\vbox to\hgt ex{\baselineskip=.2ex\vss%
      #1%
      %Format pointer underneath
      \hbox to\brd ex{\hss%
              \ea\gobble\string#1\hss}%
      }%end vbox
 }%end \pt
```

5 Disks not restricted to one digit

One could invoke `\hanoi{10}`[2] at the expense of ample time and use of paper. In order to illustrate the possibility of the macros to cope properly with disks denoted by more than one digit, the pyramid $\Big/ {9 \atop 10} \Big\backslash$ can be handled via

```
\n=2    %Number of disks
\def\ix{9}\def\x{10}%Disks
\def\I{\\\ix\\\x}\def\II{}\def\III{}
\brd=10 %Breadth of largest disk
\hgt= 6 %Height of tower
\dskhgt=1 %Height of disks
\def\\#1{\hbox to\brd ex{\hss%
  \vrule width#1ex height\dskhgt ex%
                      \hss}}%
\showtowers%Initial state
\Hanoi\I\II\III\n
```

with result



[2] Or an invoke with an even larger argument. I allowed my modestly equipped MS-DOS PC to loop for `\n` $= 1, 2, 3, \ldots$ in order to experience TEX's limits with respect to the practical upper bound for the size of the tower. A tower of 10 disks did take ≈ 1.5 minutes to TEX, with empty `\showtowers`. I had the patience to await the TEXing up to the tower of size 16, and concluded that there are no limits for reasonable usage. (Note that every increase of the size by 1 will double the time needed.) So, I don't know when my PC will crash or, to paraphrase the monks, 'when the world will end.' I also modified the macros into versions with `\I`, `\II`, and `\III` defined as token variables. I was surprised to see *that* version run 4 times slower. Whatever the value, the hanoi macros can be used as a program to compare the efficiency of TEX implementations.

6 Generalized disks

What about for example `(xyz)` as disk? Let us assume for printing that the contents of the pyramids—the strings—will do, in the implicit provided order. This can be obtained via a modified `\\` definition.

7 Example / ♠ (xyz) \

The tower can be moved, with the states printed via

```
\n=2\def\i{$\spadesuit$}\def\xyz{(xyz)}
\def\I{\\\i\\\xyz}\def\II{}\def\III{}
\brd=6 \hgt=7
\def\\#1{\hbox to \brd ex{\hss%
                              #1\hss}}%
\showtowers%Initial state
\Hanoi\I\II\III\n
```

with results

```
   ♠
 (xyz)
   I      II     III

 (xyz)           ♠
   I      II     III

         (xyz)   ♠
   I      II     III

          ♠
        (xyz)
   I      II     III
```

8 Interactivity

Downes(1991) inspired me to think about direct communication with the user. What about the modification of `\showtowers` into an appropriate `\immediate\write16{...}` command, such that the moves will appear on the screen? No previewing nor printing![3]

```
%Direct screen \showtowers
\def\showtowers{
\immediate\write16{
   \ea\gobble\string\I:   \I
   \ea\gobble\string\II:  \II
   \ea\gobble\string\III: \III}}
```

9 Conclusion

Is this just for fun? It was appropriate for 'Fun with TeX,' NTG's 91 fall meeting at Eindhoven. Furthermore, I experienced the following fundamental (TeX) programming issues

- recursion in solving the problem
- the use of the list data structure and separators to execute the list, see the TeXbook Appendix D.2
- creating and using a local loop counter[4]
- creation of a dynamic number of command names and a string of dynamic length
- generalizing the problem (not only numbers can denote disks; note that no comparison of disks is done, the ordering of the subtowers is maintained implicitly)
- direct communication on screen.

TeXnically `\aftergroup`, `\countdef`, `\csname`, `\expandafter`, `\ifnum`, `\ifx`, `\immediate\write16...`, `\loop`, `\romannumeral`, and `\string` are exercised.

The hardest thing was to get the towers aligned when formatting commands were split over several lines, due to the two-column format. Several % symbols were needed to annihilate the effect of spurious spaces, especially those created by some ⟨cr⟩'s.

It did take some time to realize the benefits of Knuth's list macros, not to say that I wandered around quite a bit.

References

[1] Downes, M.J. (1991): Dialogue with TeX. Proceedings TUG91.

[2] Graham, R.L, D.E. Knuth, O. Pastashnik (1989): Concrete Mathematics. Addison-Wesley.

[3] Knuth, D.E. (1984): The TeXbook. Addison-Wesley.

[4] Leban, B. (1985): A solution to the Tower of Hanoi problem using TeX. *TUGboat* 6, no. (3), 151–154.

[5] Pittman, J.E. (1988): Loopy.TeX. *TUGboat* 9, no. (3), 289–291.

[6] Salomon, D. (priv.comm.)

[3] Alas, no control over the format on the screen either.

[4] This was mentioned before by Pittman(1988). It is a matter of taste and programming style whether one prefers this next best to the hidden counter idea, above the use of a global counter, for counting the number of times a loop is traversed. The difference in efficiency is negligible.

Typesetting Crosswords via TEX

Kees van der Laan

March 1992

Abstract

A macro is provided for typesetting crosswords via (plain) TEX, or any TEX, which allows `\halign` use. The specification of the crossword information can be done in the WYSIWYG way,[1] and does not require `\halign` markup, just the data.

Keywords

Crosswords, games, plain TEX.

1 Introduction

This work emerged from my work on tables and FIFO—First In First Out.[2]

Hamilton Kelley (1990) published sophisticated LATEX macros for 'drawing' crosswords.

Although not a crossworder myself, it comes to mind that typesetting crosswords comes down to

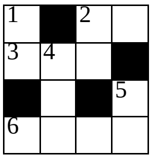
- specification of the puzzle and the solution
- providing the clues
- typesetting it all.

It is not difficult to use `\halign`. The difficulty is to keep it simple and flexible, to adopt simple conventions, and to allow natural input.

2 Examples

First the puzzle, next the clues and finally the solution to the puzzle are given as examples of use.

2.1 Puzzle



is obtained via

```
\input crw.tex
\bdata%
P*On
DEk*
*n*S
Edit
\edata\markup\data
$$\puzzletrue\crw\data$$
```

Conventions for `\bdata` $\langle\, data \,\rangle$ `\edata`[3]

- cell descriptions have to be given per 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.

I chose to provide the crossword information via the above given conventions, because it is natural and can be prepared independently. I refrained from the interactive provision of the information within TEX, because I don't know how to correct typos easily in that way.

`\markup\data` inserts the mark up necessary for the expansion of `\crw`, via redefinition of `\data`.

`\puzzletrue\crw\data` typesets the puzzle.

2.2 Clues

Is there a problem? Just columns of text, eventually aligned.[4] This subsection has nothing to do with the file `crw.tex`.

[1] What You See Is What You Get

[2] Table Diversions, to be presented at EuroTEX92, and FIFO and LIFO incognito, submitted for publication.

[3] A 'white lie,' spaces are partly allowed for crossed out cells, see the Appendix.

[4] Admitted, it is not trivial to have the clues formatted with the numbers hanging out without mark up information in the text. This can be handled by a macro with two arguments and a space as separator to distinguish the clue number from the clue text. This macro can be invoked after lines have been taken apart via `\bfifol...\efifol`. It is analogous to the nesting applied in `\markup`. This can be suitably applied within a `\valign`.

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

is obtained via

```
\smallskip\noindent
\vtop{\hsize=21ex\obeylines Across
2 Switch mode
3 Knuth
6 Prior to \TeX
}\vtop{\hsize=25ex\obeylines Down
1 Public domain
2 All right
4 All comes to it
5 Atari type}
```

2.3 Solution

`\puzzlefalse` toggles typesetting the solution, in uppercase. After `\input crw.tex`, data definition and markup, the solution

P	■	O	N
D	E	K	■
■	N	■	S
E	D	I	T

is obtained via

```
$$\def\num{}\crw\data$$
```

Redefinition of `\num`—`\def\num{}`—suppresses the typesetting of the reference numbers for the clues.

3 Programming

Crossword diagrams can be characterized by a table with 1-character elements: (marked) empty cells, crossed out cells, and letters. A cell is `\csize` by `\csize`, with height `.8\csize`. The carriage return, `^^M`, is an active character allowing WYSIWYG input. The consequence is that no &-s nor `\cr`-s are needed in the data.[5] The nice side-effect is that visual verification of the input is alleviated. The WYSIWYG provided data is transformed into marked up data ready for use within `\halign`[6] by the macro `\markup`. An inefficiency is that the *programming* of the markup is done once, while the mark up will be *inserted* whenever `\data` is invoked. The macro `\crw` typesets the crossword. The numbering of the marked cells is done automatically, row-wise and hidden.

3.1 File crw.tex

```
\let\ea=\expandafter \newif\ifpuzzle
\newcount\cnt\newdimen\csize\csize=3ex
%
\def\crw#1{\toxit{\offinterlineskip%
\halign{\bcell width0pt%of strut
\prc##\ecell&&\bcell\relax
\prc##\ecell\cr#1}%end \halign
\global\cnt0}}%end \crw
%
\def\toxit#1{\vtop{\hrule
\hbox{\vrule\vbox{#1}\vrule}
\hrule}}%end \toxit (\vtop used)
%
\def\bcell{\hbox to\csize\bgroup\vrule
height.8\csize depth.2\csize}
\let\ecell=}%TB 385, \egroup
%
\def\prc#1{\if*#1\cc\else\ifx\relax#1%
\cc\else%no crossed out cell
\ifnum‘#1=\lccode‘#1\low{#1}\else%
\cap{#1}\fi\fi\fi}%end \prc
\def\cc{\leaders\hrule height.8\csize
depth.2\csize\hfill}%end Crossed Cell
\def\low#1{\ifpuzzle\null\else\hfil%
\uppercase{#1}\fi\hfil}%end \low
\def\cap#1{\num\ifpuzzle\null%
\else\hfil#1\fi\hfil}%end \cap
%
\def\num{\global\advance\cnt1\relax%
\vbox to.8\csize{\rlap{\kern1pt%
\fiverm\the\cnt\hss}\vfil}}%end \num
%
%furthermore, in order to omit & and \cr
%in the data (natural input)
\def\markup#1{\ea\gdef\ea#1\ea{\ea%
\bfifol#1\efifol}}%end \markup
%
{\catcode‘\^^M=13 %local scope
%Pick up and processing of lines
\gdef\bfifol#1^^M#2\efifol{%
\process{#1}%process line
\ifx\empty#2\empty\let\auxl=\lr%
\else\def\auxl{\rs\bfifol#2\efifol}%
\fi\auxl}        }%end local scope
%
\def\process#1{\bfifo#1\efifo}
%
%Pick up etc of chars per line
\def\bfifo#1#2\efifo{#1%put #1 back
\ifx\empty#2\empty\let\aux=\relax%
\else\def\aux{\cs\bfifo#2\efifo}%
\fi\aux}%end \bfifo...\efifo
%
\def\cs{&}\def\rs{\cr\noalign{\hrule}}
\def\lr{\cr}
%
%next is necessary to allow spaces
%for * (except in last column)
\def\bdata{\bgroup\obeylines%
\obeyspaces\store}
\def\store#1\edata{\egroup\def\data
{#1}}
{\obeyspaces\global\let =\relax}
\endinput    %27/3/92 cgl@rug.nl
```

[5] When they are provided along with the data, the invoke `\markup\data` is superfluous.

[6] Via nested use of `\bfifo...\efifo`, see FIFO and LIFO incognito

`\crw`

The crossword is implemented as a ruled `\halign`, with the framing added separately via `\toxit`. `\offinterlineskip` suppresses the space between the rows of the table. The explicit number of cells is not necessary to specify. The counter for the reference numbers is reset at the end.

`\bcell...\ecell`

The cell size is prescribed via the size of `\hbox` and `\vrule`. For the first column `\vrule` is set to `width0pt`, for the others the default width is used.

`\prc`

The macro typesets the cell contents according to the `\data`. (`\halign` takes care of the (inserted) &-s, `\cr`-s, and `\noalign`-s.) `\if*#1` tests whether a crossed out cell has to be typeset, and if so `\cc` is invoked. For the other situation according to the case of the letter `\low`(er case letter) or `\cap`(ital letter), is invoked.

`\cc`

Crossed out cells are typeset as black cells via application of the leaders mechanism.[7]

`\low`, `\cap`

How the cell contents will be typeset depends upon the switch `\ifpuzzle`. The letters are typeset in upper case and centered. Note the invoke of `\num`.

`\num`

Generates and typesetes the reference numbers in the left upper corners.

`\markup`

The `\data` is processed per line via `\bfifol...\efifol`. The separator `^^M` splits the data. The first line is delivered in the first argument and the rest in the second argument. This splitting up is repeated recursively.

`\bfifol...\efifol`

`\bfifo...\efifo` is invoked and `\rs`, respectively `\lr`, is added. `\rs` denotes row separator, and `\lr` stands for last row.

`\bfifo...\efifo`

Inserts column separators, `\cs`.

`\bdata, \store`

These store the user provided information between `\bdata` and `\edata` in `\data`, with the carriage return as active character.

3.2 Some pitfalls

In the macro `\prc` a test is needed to reveal the case of a letter. The PASCAL-like test `\if\lowercase{#1}#1...`[8] is a pitfall; `\lowercase{\if#1}#1` and `\ifnum‘#1>96` work.

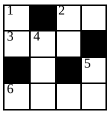
The functionality `\bdata` ⟨ *data* ⟩ `\edata` stores the user supplied information with active carriage returns. However,

```
\def\bdata{\bgroup\obeylines\gdef\data
        \bgroup}
\def\edata{\egroup\egroup}
```

is a pitfall (the `\bgroup` after `\gdef` must be an *explicit* brace!). In the examples the switch settings and the redefinition of `\num` and `\rs`, have been done *within* the math display, in order to keep the modifications local.

4 It is all in the game

Puzzle, clues and solution can be typeset all at once.



Across

2 Switch mode

3 Knuth

6 Prior to TeX

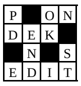
Down

1 Public domain

2 All right

4 All comes to it

5 Atari type



is obtained—after `\input crw.tex`, data definition and markup—via

```
$$\vcenter{\noindent\sevenrm\csize=3ex
\puzzletrue\crw\data\parindent=3ex
\vtop{\hsize=21ex\obeylines Across
2 Switch mode
3 Knuth
6 Prior to \TeX}
\vtop{\hsize=25ex\obeylines Down
1 Public domain
2 All right
4 All comes to it
5 Atari type}
\fiverm\csize=3ex
\puzzlefalse\def\num{}\crw\data}$$
```

The automatic placement of crosswords in the text, as ‘floating bodies,’ is not dealt with. See therefore the general `\figplace.tex`, Zalmstra and Rogers (1989).

[7] This revealed a driver bug. `\hbox to \csize {...\cc}` in the crossword was not completely black, compare ■.

[8] Jürgen Knappen communicated the elegant solution `\ifnum‘#1 = \lccode‘#1...`. Victor Eijkhout suggested among others `\ifnum \sfcode ‘#1 = 1000...` (TeXbook p.76). The elegant solution is also in his book.

5 Variations

5.1 Black or ...

The appearance of the black cells can be changed via redefinition of `\cc`.

Solutions are also typeset just with letters at the appropriate places in the diagram; without rules and black cells.

P		O	N
D	E	K	
	N		S
E	D	I	T

This can be obtained after `\input crw.tex`, data definition and markup, via

```
\def\sol#1{\vtop{\offinterlineskip%
\halign{&\hbox to\csize{\vrule%
height.8\csize depth.2\csize%
width0pt\proc##}\cr%end preamble
#1}}}%end \sol
\def\proc#1{\if*#1\else\ifx\relax#1%
\else%no black cells
\hfil\uppercase{#1}\fi\fi\hfil}%end\proc
$$\def\rs{\cr}\sol\data$$
```

5.2 No solution diagrams

Why not supply the solution similar to the clues, as columns of text?

Across	Down
2 ON	1 PD
3 DEK	2 OK
6 EDIT	4 END
	5 ST

5.3 *-symbol

It is a minor adaptation to use another symbol to denote crossed out cells.

6 Conclusion

Is this just for FUN? I hope that the approach facilitates the use of TeX for typesetting crosswords. Moreover, the worked out ideas for inputting data WYSIWYG-like and have TeX to modify them appropriately for further processing, can be applied to typesetting of other games, for example bridge and chess, and in general to typesetting of (simple) tables with natural input.

It was easier to write the macro from scratch than to understand the LaTeX macros of Hamilton Kelley. To my experience this is symptomatic for LaTeX, and its derivatives.

From the application point of view the following general issues have been dealt with

- automatic mark up
- natural (and minimal) input
- automatic and hidden numbering
- ruled tables.

TeXnically use has been made of

- filling it up (`\leaders`, `\hrule`)
- testing case of letter (`\ifnum‘#1 = \lccode‘#1`)
- *local* catcode changes for making carriage return active (`\catcode‘\^^M = 13 `)[9]
- 2-part macros and storing the data provided in between by the user
- abstraction from & and `\cr`
- parameterizing via a switch and redefs
- testing for empty argument (`\ifx \empty #2 \empty...`)
- nested if-s
- replicator mechanism in preamble
- parameter separation
- FIFO recursion.

The author claims that crosswords can be typeset easily and of high-quality via TeX.

All the nitty-gritties put in reminded me of DEK[10]

> ‘Yet dozen small refinements add up to something that is important to me, and I think such refinements might prove important to other people as well.’

7 Acknowledgements

I like to thank Jürgen Knappen for providing the elegant test for the case of a letter. David Salomon is kindly acknowledged for the stimulating (e-mail) discussions.

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 (submitted for publication): FIFO and LIFO incognito.

[4] Salomon, D (priv.comm.).

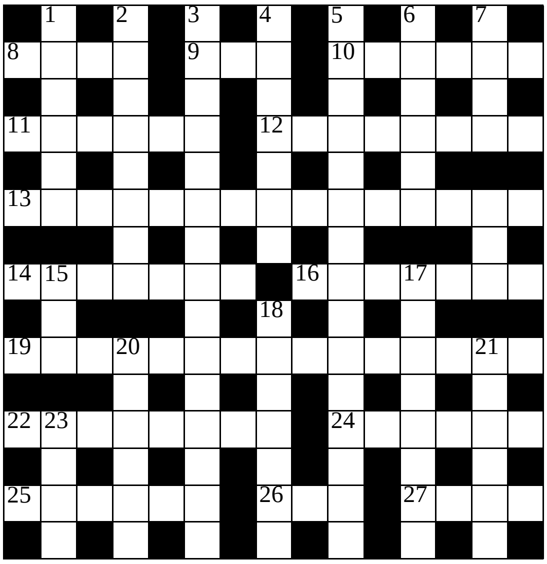
[5] Zalmstra, J, D.F. Rogers (1989): A page make-up macro. *TUGboat* 10, no. (1), 73 – 81.

[9] Also touched upon is to allow spaces for *’s, except at the line ends. This entailed making spaces active.

[10] Mathematical Typography, Bulletin of the AMS, 1, 2, 1979, p.345.

Appendix

Hamilton Kelley's puzzle[11]



is obtained—after `\input crw.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\markup\data
$$\puzzletrue\crw\data$$
```

The above shows that the macros also take spaces for crossed out cells, except for those in the last column. It seems to be impossible to get hold of spaces at line ends in TEX, because TEX deletes endspaces, to accommodate for some computer series which fill up lines with trailing spaces. I could circumvent the S-state, where multiple spaces are skipped, TEXbook p.46.

The above is the next best to complete natural input for crosswords. If the spaces interfere with other parts of the document, the last line `{\obeyspaces...}`, of `crw.tex` can be deleted, and no spaces are accounted for in the crossword.

[11] Without BHK's clues, see for those his article.

LADIES and LATEX– II

Theo A. Jurriens

Kapteyn Astronomical Institute
P.O. Box 800
9700 AV Groningen
The Netherlands

taj@rugr86.rug.nl

Abstract

In dit artikel wordt mijn ervaring als LATEX-docent, vraagbaak voor secretaresses beschreven. Er zijn twee bronnen van problemen: organisatie en TEX-techniek.

1 Inleiding

Sinds een aantal jaren ben ik LATEX vraagbaak voor het secretariaat van de Afdeling Sterrenkunde van de Rijksuniversiteit Groningen. Ook de astronomen, veelal geen dame's, behoren tot mijn 'klantenkring'. In deze bijdrage wil ik mij richten op het gebruik van LATEX door secretaresses. Het is niet alleen gebaseerd op de ervaringen in Groningen en met de dames aldaar maar ook ervaring opgedaan tijdens diverse cursussen LATEX in Nederland en ver daar buiten (zie Appendix voor overzicht). Het artikel [1] geeft een overzicht van LATEX gebruik in Groningen.

De problemen inzake introductie en gebruik LATEX zijn twee-ledig: enerzijds zijn ze van organisatorische aard, anderzijds van technische en ook LATEX/TEXnische aard. Beide worden in deze bijdrage belicht.

2 Organisatie

In deze bijdrage beschouwen we de opdrachtgever van de secretaresse [1] als de klant, de secretaresse als de winkelier. Het LATEX/TEX-gedrag van de klant dwingt de winkelier tot het leren van LATEX: de klant krijgt ook steeds meer LATEX klanten: de grote uitgevers zien TEX en zeker LATEX tot **de standaard** voor de komende jaren. In de meeste gevallen leert de winkelier LATEX: de praktijk leert immers dat 95% van de klanten dan ook wel tevreden zijn. Maar ja de 5% ontevreden klanten kunnen het leven van de winkelier **zuur** maken. De winkelier moet altijd klant vriendelijk zijn, immers klant is koning. Daarnaast is de winkel altijd open en moet er uit 'voorraad' geleverd worden. Dat betekent in de praktijk dat de secretaresse nauwelijks tijd krijgt om LATEX te leren, laat staan om er rustig mee te oefenen.

Mijn cursussen geef ik dan ook zelden op de werkplek, wat uiteraard zijn nadelen heeft. Tenzij ik de garantie krijg van de 'klant' dat hij de winkelier enige dagen met rust laat. Genoemde cursus duurt minimaal twee dagen. Daarnaast waarschuw ik de klant dat hij of zij de winkelier leert om met haar nieuwe produkt om te gaan. Auteurs kunnen niet verwachten van een beginnend LATEX typiste dat zij het gelijk voor 100% beheerst. Dat doet de klant ja ook niet, zeker in het begin kan het voorkomen dat de klant meer van het produkt weet dan de winkelier. Ook houd ik altijd een pleidooi dat de winkelier beter beloond wordt. Goed TEX of LATEX vereist meer inzicht en vaardigheden van de winkelier dan zijn WP-collega.

Als de winkelier moderniseert van zijn WP-winkel naar zijn LATEX-winkel heeft dan enkele nadelen. Een LATEX-starter is veel beter, laatst genoemde is niet gehandicapt door WP-kennis. De bediening van de WP-winkel is veel vriendelijker voor de winkelier: immers de magische `F3` toets geeft alle informatie over de winkel. En veel winkeliers zijn opgeleid tot WP-handelaar. Zover mij bekend zijn er op erkende secretaresse opleidingen nog geen cursussen geavanceerd document opmaak. Kortom in veel gevallen moet de winkelier zijn TEX of LATEX adhoc leren. Maar gelukkig is er nog een vertegenwoordiger die af en toe de winkelier kan trainen. Maar ja als er problemen zijn, is hij niet altijd binnen handbereik. En de klant is er toch niet om de winkelier te helpen?

'Gouden regels' voor LATEX introductie op secretariaatsniveau:

1. geef de ruimte om LATEX te leren,
2. geef de ruimte om LATEX te oefenen, dus geen morgen af klussen in het begin,
3. iedere secretaresse een LATEX boek,

[1] de auteur is er zich van bewust dat ook mannen uiterst bekwaam als secretaris fungeren, dit artikel is ook zeker niet vrouw-onvriendelijk bedoeld.

4. een printer, om de mooie resultaten af te drukken, binnen handbereik,
5. overleg met de secretaresse voor het inleveren van het te typen manuscript,
6. vertrouw op de LATEX ervaring van de secretaresses en wees een plezierige 'klant',
7. gebruik LATEX ook voor het simpele werk: het niet-formule werk. Op deze manier wordt de 'winkelier' gedwongen één standaard te hanteren,
8. zorg voor de juiste beloning.

3 TEX-techniek

Eerder is genoemd dat WP een gebruikersvriendelijke tekstverwerker is. Voor secretaresses, vers van de opleiding is het de standaard. Voor een efficiënt TEX of LATEX gebruik is een intelligente, gebruikersvriendelijke editor gewenst: een menu gestuurde emacs, een emacs á la Beebe of een speciale optie in WP: 'saven as TeX'. Overigens de nieuwe versie van K-Talk (de WP naar TEX conversie) schijnt goed[2] te werken. Sommige secretaresse's gebruiken WP als editor, werkt goed maar het gevaar bestaat dat de files per ongeluk niet als ASCII bewaard worden. In *TUGboat* is eerder bericht over WP als intelligente LATEX-editor. Een ander 'gevaar' is dat als WP aanwezig is dat het gebruikt wordt voor het simpele werk.

Op een gegeven moment ontstaat de behoefte om zelf definities of eigen styles files te maken: pure TEX kennis is dan nodig maar niet altijd aanwezig. Bestaande style files zijn onleesbaar en bovendien is het LATEX boek voor dit doel onbegrijpelijk. Het is een veel gehoorde klacht: in het boek van Lamport kun je niets vinden. Het wordt versterkt door onvoldoende kennis van de Engelse TEX-termen.

Het maken van serie-brieven (mailing) is voor een secretaresse niet voor de handliggend. Een lokale TEX-specialist is nodig. Evenzo geldt dit voor het uptodate houden van de software. Een secretaresse kan veelal niet verantwoordelijk worden gesteld voor systeemonderhoud.

De eerder genoemde 'Gouden regels' kunnen we als volgt aanvullen:

1. zorg voor goede apparatuur en onderhoud van software,
2. een goede editor,
3. een lokale TEX of LATEX vraagbaak.

Naast deze regels, tips is er nog een taak voor de diverse TEX User Groups. Maak een internationale dictionary van TEX, LATEX termen. Een goed boek, in de eigen taal, is er nog niet voor deze doelgroep[3]. Adresbeheer à la BIBTEX, serie brieven. Een problemen rubriek van en voor deze doelgroep in de nationale TEX organen! In de volgende MAPS de eerste aflevering: *problemen met dames.*

References

[1] Theo Jurriens, LADIES *and LATEX*, MAPS 91.2, 102–104.

[2] Theo Jurriens, *TEXniques in Siberia*, MAPS 91.2, 87–90.

LATEX-cursus ervaring:

- Secretariaat Sterrenkunde, Lab. voor Ruimteonderzoek Groningen,
- Wiskunde en Informatica, Rijksuniversiteit Groningen,
- Vakgroep Econometrie, Rijksuniversiteit Groningen,
- Kernfysisch Versneller Instituut, KVI Groningen,
- Vakgroep Econometrie, Katholieke Universiteit Brabant,
- Vakgroep Natuurkunde, Universiteit van Amsterdam,
- Rekencentrum en Universiteit Novosibirsk.

[2] niet zelf getest

[3] een reeds lange gekoesterde wens van me.

000451 SS Cas
Mira HIP

560.4	13.0	FJH
572.5	13.3	FJH
594.5	12.5	FJH

000928 UW And
Mira

533.4	11.7	FJH
556.3	12.7	FJH
570.5	13.7	FJH

001046 X And
Mira

532.4	10.9	FJH
551.4	9.9	FJH
572.4	9.1	FJH
596.4	9.1	FJH
611.4	9.8	FJH

001726 T And
Mira

557.4	10.6	FJH
572.4	9.5	FJH
596.4	8.2	FJH
614.4	8.5	FJH

001755 T Cas
Mira HIP

596.4	9.0	FJH
614.4	9.1	FJH

001838 R And
Mira HIP

540.5	9.1	FJH
557.3	9.8	FJH
572.4	10.3	FJH
596.3	10.9	FJH
611.4	11.2	FJH

002725 A TU And
Mira HIP

557.4	11.5	FJH

003162 TY Cas
Mira

576.3	13.5	FJH
594.5	11.5	FJH

003179 Y Cep
Mira

551.4	13.0	FJH
570.5	13.6	FJH
590.6	14.0	FJH

004047 U Cas
Mira

540.4	12.7	FJH
550.4	13.0	FJH
557.3	13.7	FJH
572.5	14.2	FJH
596.3	:15.5	FJH

004132 RW And
Mira

533.5	15.3	FJH
596.3	:15.5	FJH

004435 V And
Mira

557.3	14.1	FJH
569.5	14.5	FJH
596.3	14.8	FJH
615.3	14.3	FJH

004533 RR And
Mira

533.5	15.0	FJH
557.3	13.8	FJH
569.5	12.1	FJH
596.4	10.9	FJH
611.4	9.8	FJH

004746 A RV Cas
Mira

533.5	15.5	FJH
557.3	15.0	FJH
601.5	12.6	FJH
611.4	11.7	FJH

004958 W Cas
Mira HIP

556.6	:10.1	KKP

005840 RX And
UGZ

532.42	14.0	FJH
533.48	13.7	FJH
536.43	14.0	FJH
540.44	11.5	FJH
545.38	12.0	FJH
551.34	13.5	FJH
556.35	13.0	FJH
557.32	13.4	FJH
558.37	13.1	FJH
559.38	13.5	FJH
560.35	13.6	FJH
569.46	12.3	FJH
570.35	11.2	FJH
572.39	11.5	FJH
574.43	11.6	FJH
575.34	11.6	FJH
576.33	12.2	FJH
594.44	13.7	FJH
596.35	13.7	FJH
615.28	13.9	FJH

010621 A X Psc
Mira

536.4	14.5	FJH
558.4	13.9	FJH
570.5	13.5	FJH
596.4	12.9	FJH

010937 FO And
UG

533.47	14.9	FJH
536.45	<15.4	FJH
572.46	14.6	FJH
596.34	<15.4	FJH
615.28	14.2	FJH

010940 U And
Mira

540.5	9.6	FJH
557.4	10.2	FJH
569.5	11.1	FJH
594.4	11.9	FJH
614.4	12.3	FJH

011041 A UZ And
Mira

533.5	15.1	FJH
557.3	14.0	FJH
569.5	13.9	FJH
594.4	13.2	FJH
614.4	12.0	FJH

011055 A VZ Cas
Mira

560.4	10.6	FJH
611.4	11.0	FJH

011208 S Psc
Mira

540.5	11.0	FJH
557.5	11.8	FJH
570.5	12.2	FJH
596.3	12.7	FJH

011712 U Psc
Mira

536.4	14.2	FJH
558.4	14.4	FJH
570.5	13.6	FJH
596.3	12.2	FJH

012020 RX Psc
Mira

536.4	:15.1	FJH
596.3	14.5	FJH

012031 TY Psc
UGSU

594.45	12.1	FJH
596.34	12.4	FJH

012502 R Psc
Mira

536.5	13.8	FJH
559.3	13.9	FJH
570.5	13.9	FJH
596.3	13.1	FJH

012746 SX And
Mira

556.4	10.6	FJH
575.3	11.1	FJH
594.5	11.5	FJH
611.4	12.0	FJH

013050 KT Per
UGZ

536.44	12.4	FJH
540.46	14.4	FJH
556.35	12.2	FJH
557.32	11.9	FJH
558.36	11.8	FJH
559.38	11.8	FJH
560.35	12.0	FJH
572.46	14.4	FJH
575.35	12.4	FJH
576.33	12.4	FJH
590.52	11.9	FJH
601.46	11.9	FJH

013238 RU And
SRa

536.5	12.0	FJH
556.4	11.7	FJH
575.3	11.9	FJH

013338 Y And
Mira

532.4	14.2	FJH
540.5	14.0	FJH
551.3	13.2	FJH
569.5	11.3	FJH
594.5	9.9	FJH
611.4	9.2	FJH

013937 AR And
UGSS

532.42	14.2	FJH
533.47	15.3	FJH
556.36	12.5	FJH
557.32	12.8	FJH
558.36	13.5	FJH
559.39	<15.0	FJH
572.36	12.3	FJH
574.43	12.7	FJH
575.36	14.0	FJH
594.45	11.9	FJH
596.32	12.2	FJH
601.46	12.7	FJH

015254 U Per
Mira HIP

533.4	8.3	KKP
536.4	8.5	JOJ
556.3	8.2	JOJ
556.6	8.7	KKP
601.3	9.1	KKP
601.4	8.4	JOJ

015457 V666 Cas
Mira

556.4	11.2	FJH
576.3	11.2	FJH

015912 S Ari
Mira

533.5	13.8	FJH
558.4	14.5	FJH
570.5	14.6	FJH
596.3	:15.2	FJH

020227 Z Tri
Mira

532.5	14.7	FJH
540.5	14.7	FJH
596.3	13.7	FJH

020356 UV Per
UGSS

613.51	12.6	FJH
614.40	12.7	FJH
615.23	12.7	FJH

020657 A TZ Per
UGZ

532.50	12.9	FJH
536.44	13.8	FJH
540.46	14.3	FJH
551.35	12.8	FJH
557.33	13.9	FJH
558.36	14.2	FJH
559.39	14.2	FJH
560.48	13.7	FJH
570.35	13.5	FJH
572.47	13.8	FJH
575.36	13.7	FJH
590.51	13.6	FJH
596.46	13.6	FJH
601.46	12.7	FJH
615.27	13.4	FJH

021024 R Ari
Mira HIP

540.5	12.5	FJH
551.4	12.4	FJH
575.5	10.5	FJH

021143 A W And
Mira HIP

540.4	13.1	FJH
551.3	12.7	FJH
569.5	12.4	FJH
594.5	11.8	FJH
611.4	10.7	FJH

021281 Z Cep
Mira

536.5	14.1	FJH
551.4	13.5	FJH
570.5	11.9	FJH
581.3	11.6	FJH
590.6	11.8	FJH
613.5	12.3	FJH

0214-0 3 Mira
Mira HIP

536.5	4.0	SAQ
536.6	4.0	BMU
596.3	6.1	SAQ

021558 S Per
SRc HIP

556.4	12.0	FJH
557.3	11.8	HIL
576.3	12.2	FJH
594.5	12.2	FJH

0220-0 0 R Cet
Mira HIP

536.5	9.2	SAQ
601.3	8.3	SAQ

022150 RR Per
Mira

540.5	11.5	FJH
556.4	11.7	FJH
575.4	12.2	FJH
590.5	12.7	FJH

022980 RR Cep
Mira

536.5	14.1	FJH
570.5	13.1	FJH
581.3	12.5	FJH
590.6	11.8	FJH
613.5	11.2	FJH

023133 R Tri
Mira HIP

533.4	9.7	KKP
540.3	10.6	HIL
557.3	11.6	HIL
557.5	11.8	FJH
570.5	12.0	FJH
594.5	12.3	FJH

030226 Z Ari
Mira

533.5	13.8	FJH
559.4	:14.4	FJH

030514 U Ari
Mira

540.5	10.5	FJH
559.4	11.6	FJH
596.3	13.6	FJH

031170 V667 Cas
Mira

572.5	10.5	HIL
596.5	9.4	FJH
600.5	9.8	HIL

032043 Y Per
Mira

533.4	9.1	KKP
556.6	9.8	KKP
580.5	10.1	KKP
584.5	10.3	KKP
597.4	9.7	KKP

Book reviews[1]

Nico Poppelier, Amy Hendrickson

March 1992

LATEX for Everyone,
Jane Hahn,
1st edition, Personal TEX Inc. 1991,
softbound, 346 pages

Writing a book is hard work. It can also be rewarding work – if the readers are satisfied with the book. In comparison, writing a review about a book is easy: in a few paragraphs you criticize what it tooks years to write. Nevertheless, the readers deserve an honest review, so I won't hide the fact that in my opinion the first book reviewed here is less than what it could have been. This book, *LATEX for Everyone* by Jane Hahn, is published by Personal TEX, Inc. (PTI), and which will replace *LATEX, a Document Preparation System* by Leslie Lamport in the PC-TEX packages that PTI sells.

Surely, Lamport's book leaves a lot to be desired as an introductory book. For this purpose, you need a book with a clear expository style, a sufficient number of examples and well designed exercises. On the surface, it looks as if *LATEX for Everyone* could have been such a book, since it has a clear 'if you want this, do that' way of explaining, it has summaries at the end of all sectional units, and lots of exercises.[2] Unfortunately it falls short of being a good introduction: it shows structural flaws, it contains a substantial number of mistakes, and it explains several parts of LATEX confusingly or not at all.

Structure

Chapter 2 introduces the basic commands of LATEX, and it also tells you how to adjust line spacing, margins, paragraph indentation, and footnote spacing – I will come back to this in a minute.

Chapter 3 is an odd mixture of things: it explains about document styles, typefaces and typeface sizes, sectioning commands, symbolic references, hyphenation, lists, formulas, accents, and headers and footers.

Chapter 4 deals with mathematics, but the environments for displayed equations were treated in chapter 3. Chapter 5, *Rows and Columns*, discusses `tabbing`, `tabular`, `array` and `eqnarray`. There are two problems with this arrangement of material.

1. The information on mathematical formulas is spread over three chapters.
2. `array` is used in chapter 4 on pages 93 and 99, but is not explained until later on, on page 128.

Chapter 6, *Customization*, treats page and line breaks, centering, vertical and horizontal space, lengths and boxes. This is followed by a chapter on floating objects and one on preparing large documents. In my view, chapter 6 should have been put after chapters 7 and 8, and combined with parts from chapter 3 in a chapter on influencing the layout.

Chapter 7 contains a lot of useful information about floating tables and figures, but it could have been written more concisely I think. And, like other authors of books on LATEX – see some of my earlier reviews – Ms. Hahn does not clarify what `table` and `figure` are, namely 'envelopes' for floating figures and tables.

Furthermore, the book contains seven appendices. Appendices A and B, on user-defined commands and counters respectively, contain lots of useful information with instructive examples. Appendix C, on style parameters, is also a nice appendix, but it lacks the page-layout and list-layout diagrams, which are by now familiar to most LATEX users.

Appendix D treats the `picture` environment. Appendix E, *Errors*, is a particularly good appendix, with lots of examples. I missed one thing in this appendix: what happens when you forget the required argument of `\begin{thebibliography}`?

Appendix F gives examples in the form of question and answer, and is one of the best parts of the book!

Appendix G 'discusses' SLITEX in twelve (sic!) lines. The page on which it is printed can just as easily be torn out of the book, since all it tells the reader is that SLITEX is a program similar to LATEX, designed for creating slides, and with commands different from those of LATEX. If the reader wants to know more, he

[1] To be published in TUGboat 13.1, 57–59 (1992).

[2] I should add that the answers to the exercises are given in small print below the questions.

or she is advised to print and read `slides.tex` and `local.tex`.

Finally, the index is awkward to work with: all environments must be looked up under the main entry 'environment', and all commands under the main entry 'commands'. Strangely, the entry 'commands' is followed by 'captions', '`center`', 'comment', . . .[3] My preference would be to list, e.g., '`picture`' environment between 'picture' and 'placement', as in the LaTeX User's Guide, or to have a separate command index.

My main criticism is that the structure of *LaTeX for Everyone* does not reflect the philosophy behind LaTeX – like most other books on LaTeX unfortunately. Chapters 2–4 of *LaTeX, a Document Preparation System* by LaTeX's creator Leslie Lamport mostly explain about those features of LaTeX that are related to logical structure of a document. Only in chapter 5 does he discuss those features that are more related to the visual structure of a document.

By contrast, Ms. Hahn continually mixes structure commands with layout commands.

An example: in almost every chapter Ms. Hahn introduces a command that accepts the `\\` command, and every time she explains what `\\[...]` does. If she had moved this to a separate appendix on layout changes, this would reflect the philosophy of LaTeX, and it would make the exposition much clearer.

Another one: in section 3.10.1 she gives this example

```
\begin{itemize}
\item [$\heartsuit$] potatoes
\item [$\heartsuit$] celery
\item [$\heartsuit$] frying chicken
\item [$\heartsuit$] milk
\end{itemize}
```

immediately after she has introduced the `itemize` environment. First of all, this can be done much simpler with a `\renewcommand` of `\labelitemi`. Secondly, this sort of example really belongs in a separate chapter on layout changes.

Errors

This review column does not provide the space required for an extensive summary of all errors in *LaTeX for Everyone.* Instead, I will mention a few interesting ones.

1. The author confuses the document style `book` with the abstract class of documents that can be called 'book'. Furthermore, to confuse the reader she introduces a new term, 'style guide', as a synonym for 'document style'. She also confuses LaTeX with its standard document styles (p. 69–70)
2. On page 42 she calls TeX's 'usual' typeface, Computer Modern, Times Roman.
3. On page 88: 'A super- or subscript that is an English word should be set in roman type'. Is this not the case for mathematical texts in French or Dutch?
4. An explanation of `*{n}{cols}` is missing in all places where `tabular` is treated (pages 127 and 288).
5. 'You should get into the habit of typing names as follows: . . . `J.~S.~Bach`' (page 142). Not true, since it depends on the particular typographical convention one uses: in common usage the space between 'J.' and 'S.' is omitted.
6. A table in section 6.6 suggests that LaTeX does not understand the following units of length: `dd`, `cc`, `bp` and `sp`, which the basic TeX program, and therefore LaTeX, an extension, understands.
7. In section 6.7, the author uses `\makebox` to get an alignment!
8. On page 164, Ms. Hahn writes that
   ```
   \oddsidemargin=0in
   \textwidth=6.5in
   ```
   results in a right margin of 1 inch. This happens sometimes, but only if you use American letter size paper!
9. The 'default order of preference' for figure placement is `[bthp]`', according to the author, which is wrong, since this default is given by the document style, for example `[tbp]` in `article`.
10. On pages 194–197 Ms. Hahn suggests producing an index by sorting the entries in the `.idx` file in your editor, manually changing the `\indexentry` commands into `\item` and so forth, and then combining multiple entries into one. I find this appalling advice, with index programs such as MakeIndex available.
11. Similarly, in section 8.5 there is no mention of BibTeX.
12. The command `\setlength` is discussed in the main text, whereas `\newcommand`, `\renewcommand` and `\newenvironment` are treated in the appendices. In my view, the latter are more important, because they make typing easier or can clarify the structure of a document. A separate appendix on layout changes would be an appropriate place to discuss `\setlength`.
13. A discussion of `\newtheorem` is completely missing.

Besides this, Ms. Hahn sometimes suggests bad typography. For example a tall formula, an integral in display style, in text. Shouldn't authors of books on TeX keep traditional typographical rules of thumb in mind?

Conclusion

On the whole, *LaTeX for Everyone* is an unsatisfactory book. It has the potential of becoming a good book, in a revised edition, if the structural flaws are solved and all the errors are removed.

[3] Probably because the index was generated as explained on pages 194–197 of the book – see further on.

The author considers math LaTeX's strongest feature, a position I disagree with strongly: its main merit is document structuring. Math is a TeX feature, and LaTeX does not *add* new math capabilities: it presents them in a structured and sometimes more user-friendly way. If Ms. Hahn had recognized the key role of document structuring in LaTeX, she would probably have written a different book.

A final remark: the publisher chose to have the book produced from 2000 dpi camera-ready copy, which is the high quality output a book on TeX, made by TeX deserves. Unfortunately, the typeface Computer Modern was used, and the layout is the standard `book` style. That TeX can produce 'masterpieces of the publishing art',[4] using other fine typefaces and a layout created by a professional designer, is shown too rarely – an exception is Victor Eijkhout's recent book *TeX by Topic.*

Nico Poppelier
`n.poppelier@elsevier.nl`

Practical SGML,
Eric van Herwijnen,
1st edition, Kluwer Academic Publishers 1990,
softbound, 307 pages

'A review of a book on SGML in the columns of *TUGboat*?' some of you may wonder. What does SGML have to do with TeX? Well, nothing, but since the term SGML has surfaced often in *TUGboat* and on the TUG conferences the past years[5], I thought a review of an SGML book could be worthwhile.

Practical SGML is one of the best books on SGML currently available. To be absolutely honest, there are not many books on SGML – yet – but this book is the only one so far with 'many helpful hints and ideas on developing SGML, applications and discussions of the current software written to be conforming to the ISO standard', as is written in the foreword of the book. This is indeed a book about practical SGML!

The book is divided into three parts. Part I, *Getting started with SGML*, is an introduction to SGML. It explains what a document type definition or 'dtd' is, what the role of the dtd in the processing of the document is, and what steps are necessary to create and process an SGML document.

Part II is intended for document managers or programmers, and explains SGML in more depth. Some of the topics discussed in this part are: formal aspects of the language SGML, distinguishing data characters from markup, and the reference concrete syntax.

Part III is about SGML implementations and should be read by everyone who has to install and maintain an SGML software system. Mr. Van Herwijnen discusses what components are usually found in such a system, how to create SGML documents, how to convert SGML documents into documents that can be processed, for instance to get output on paper, or in order to store information in a database. He also gives some examples of SGML parsers.

The book also contains five appendices. Appendix A contains the answers to the exercises in the book. In appendix B Mr. Van Herwijnen tells how he wrote *Practical SGML* using SGML, and in appendix C he even gives the complete document type definition for his book.

Appendix D is a short appendix, in which the author gives common SGML definitions for use with TeX. Finally, appendix E contains useful advice on how to read the ISO standard (8879) in which SGML is defined.

At the end of the book we find a glossary and an index, and throughout the book the author gives lots of valuable references to existing literature on SGML and related topics.

Mr. Van Herwijnen, is leader of the text processing section at CERN, the European Laboratory for Particle Physics in Geneva, Switzerland. SGML is one of the important tools in the text processing section at CERN, which probably explains the high quality of *Practical SGML*: it was written by someone who has extensively used SGML in practice. Since no prior knowledge of text processing or publishing is required to understand what is written in *Practical SGML*, I can highly recommend it to anyone who is interested in this subject.

Nico Poppelier
`n.poppelier@elsevier.nl`

TeX by Topic, A TeXnician's Reference,
Victor Eijkhout,
1st edition, Addison-Wesley Publishing Company, 1992
softbound, 307 pages

Victor Eijkhout has definitely done the TeX community a service by presenting us with his book, 'TeX by Topic', published by Addison-Wesley. I recommend it highly to everyone who has had some acquaintance

[4] The last line of the last chapter of *The TeXbook.*

[5] See for example the proceedings of the 1991 TUG conference.

with TEX and who would like to 1) understand the basic mechanisms underlying TEX processing and 2) wants a concise TEX reference source.

The key to this book's success is the considerable effort and thought that went into the overall conception and in the care taken in transforming that concept into a systematic and thorough development of each topic presented.

It is a daunting task to write about TEX, since to describe any topic thoroughly inevitably includes defining its component parts, which can launch the author into another topic to the confusion of the readers. It is a task that must be approached with care.

This organizational problem can be seen even in the TEXbook, though one hesitates to complain. We TEXies have all spent hours poring over the TEXbook, enjoying its lovely writing, its complex concepts, and its illuminating examples. One cannot help admiring it and its author, of course. Still, it must be admitted, when a macro writer is struggling to find some particular bit of information, a quick definition, a nudge in the right direction, he or she can find the TEXbook index thoroughly unhelpful and the bit of information wanted scattered over many different pages in many different chapters.

'TEX by Topic,' on the other hand, excels in organization, making it a most helpful adjunct to the TEXbook. The basic plan is to break the whole subject of TEX commands and processing into about 40 topics, each discussed in its own chapter. Each chapter begins with a list of control sequences relevant to that chapter, followed by a brief explanation of the theory behind that topic, then brief remarks and examples. The chapters in the book are organized into three parts: chapters on basic mechanism; text treatment and math; and finally, output and aspects of TEX's connection to the outside world.

The book is meant as a reference source, not a tutorial. It assumes that the reader has a general grasp of TEX vocabulary and syntax. It is brief and concise and information dense.

It includes detailed explanations of the mechanisms underlying TEX's workings and TEX programming techniques. In the last part of the book is found a description of the differences between TEX Version 2 and Version 3, a 13 page glossary of TEX primitives, a bibliographical list of references, character tables, math symbols, a list of examples. The book ends with a 3 page Index by command and a 2 page Index by topic, each with sending readers to *one* particular page for information on each entry.

Evaluation:
I have found the book useful several time already, and I have been using TEX for the last nine years. Even if you have been using TEX for a number of years you may not have had occassion to use a particular command and using 'TEX by Topic' may save you valuable hours by quickly defining or demonstrating the bit of information you need. In the event that you must also consult the TEXbook, having two explanations rather than one for a given topic can be helpful.

I think my complaints about the book relate to its brevity which is also one of its virtues. 'T. by T.' tends to be more theoretical than practical, and it is somewhat sparse with examples. There are many times when it might well be helpful to readers to explain 'why' a particular construct should be used, as well as mentioning that it exists.

For instance, the strut is mentioned on p. 213. First, it is explained somewhat inaccurately by saying it is defined statically in Plain TEX as a `\vruleheight 8.5pt depth3.5pt width 0pt`.

Actually the definition of the `\strut` command only asks to copy the strut box. The strut box has been separately set to include a rule of those dimensions. When making up font families one need not change the definition of `\strut` but only the contents of the `\strutbox`. The `\strut` command itself is not static. The quick explanation given 'TEX by Topic' is understandable as an effort in extreme brevity but could be slightly misleading if taken literally. Secondly, the strut is mentioned only as being used in tables. How else can or is it used? Perhaps this question is considered to be too basic, but the strut is found in many other macros and its use could be explained further.

A similar example is found in the brief mention of `\everycr`. 'T. by T.' mentions that it is included after every nonredundant `\cr`, and defines it in the list of primitives. But how might the TEXnician make use of it?

Or, in the explanation of '`\let`' and '`\futurelet`' which are at least as clear as those found in the TEXbook, perhaps it would be helpful to let users know why it is advantageous to use those constructions instead of the roughly analogous '`\def`' and '`\expandafter`.' But perhaps that is a different book, a book on macro writing.

In the end, all this is mere quibbling. There is a wealth of well organized material included in this books' 300 pages. I am delighted to have 'TEX by Topic' at my elbow as I work, and if you are a serious TEX user, you will be too.

Amy Hendrickson,
TeX nology Inc.
`amyh@ai.mit.edu`

Spivak's Œvre

Kees van der Laan

March 1992

Abstract

Spivak's The Joy of TeX and LAMS-TeX— The Synthesis, are discussed.

1 Introduction

To my knowledge Spivak's work comprises the books The Joy of TeX, LAMS-TeX—The Synthesis, and his Wizard's manual. Next to DEK's books these are the best documented books/manuals of $\langle X\rangle$TeX implementations, I have seen. As should be the case with books which have passed some barriers the *quality* of the material is not the issue,[1] but more the relevancy, the intelligibility and the functionality compared to plain. Some numbers: plain is in the TeXbook 483p. and manmac comprises 12p, 'The Joy of TeX' another 290p. (no macro listings), LAMS-TeX, again 290p. (no macro listings either).

1.1 Dependencies

AMS-TeX has been commanded by the American Mathematical Society (AMS). It is not a proper extension to plain. DEK supervised the project. It is another TeX variant, like LaTeX is. LAMS-TeX builds upon AMS-TeX and provides LaTeX functionalities except for the picture environment, Spivak claims that it is less verbose than LaTeX. The file `amstex1.tex` is needed to run LAMS-TeX. LAMS-TeX's table modules seem to be independent of AMS-TeX; I don't know whether the commutative diagram macros are. All the programs are in the public domain.

2 AMS-TeX—The Joy of TeX

The book consists of 3 parts: Starters ($\approx$ 50p), Main courses ($\approx$ 70p), and Sauces & Pickles ($\approx$ 75p), complemented with 10 appendices ($\approx$ 75p).

It is meant as an independent extension to TeX, and to the TeXbook. A consequence is that no references to the TeXbook are made, which I pity. The design is at least 10 years old. Moreover, the introduction says that 'You can remain blissfully ignorant of the complicated rules that typesetters have developed for the proper setting of mathematics formulas—TeX knows them all.' Well, that is not true. An author or typist can*not* remain blissfully ignorant, as is among others proved by the example on the next pages: explicit kernings and context dependent parentheses are used. Knowledge of typesetting in general and typesetting of mathematics in particular, remains necessary. A reference to Swanson (1986) should have been made. Moreover, I don't consider the approach realistic: AMS-TeX users should at least be TeX users. The abstraction from formatting into procedural mark up—although to be recommended in general—goes so far that the basic concepts of TeX: boxes, glue and penalties, are not explained. It is even stated at p123 '`\h(v)box` isn't a control sequence that you are ever supposed to use—it is a control sequence that TeX uses internally in all sorts of important ways.' I don't agree with the one-sidedness of the made choices, especially because it is overlooked that typists also need to get the TeXscript correct, that is, it has to pass the parser, and proofs have to come out, that is with use of the right fonts. There is no doubt about it that the error messages, and unawareness of the font selection schemes will cause trouble. Now and then the book says: 'AMS-TeX tries . . . ,' while it should have been 'TeX tries . . . '.

2.1 First part: Starters

Basic AMS-TeX processing, and the use of `\amsppt` style (the preprint style), constitute part 1. A nice getting started section. Some quibbles, however. Especially in relation to LAMS-TeX, the use of double quotes for quotations makes me realize that people change minds, and Spivak in particular.[2] What I missed with respect to spacing is the good habit to terminate a number by a space or `\relax`. Furthermore it is a white lie (p5) that curly braces can't be replaced by control sequences. Indeed not *all* occurrences can. I presume that `\bgroup` must be kept hidden for the user as well? In the subsections about spacing I missed the example `\TeX` itself. Spaces, trailing and the spurious ones, remain puzzling. That replacement texts *don't*

[1] And indeed let there be no doubt about it: the quality is very good!

[2] The more so when in the LAMS-TeX manual double quotes are used for a quotation, p187.

neglect spaces is well demonstrated among others by the exercises 19.3, .26, and .27.

2.2 Second part: Main Courses

This section constitutes what it is all about: to simplify the inputting of math.

Nowhere it is proofed or made acceptable in some way that the provided macros are better or easier to use. The macros are mainly different from plain's. In general terms the section is well-done. In exercise 8.10, I don't like the use of the multiplication dot.[3] When it is mentioned that no `\par`-s are allowed in display math mode, a note about blank lines would have been appropriate too.[4] In section 10 it is stated that the instructions ^, and _ apply only to the next single character. A white lie: it applies to the next token or group, in plain at least.

I don't like to be carried away by TeX's power. To become a bit loose about the good habit to choose as simple representations as possible, because of TeX's power is counter productive with respect to getting across what an author is up to. This holds for division representations, complicated exponents without use of `\exp` (there is also the spacing pitfall in the exponents), and complicated 'limits' to summation and integration symbols. Swanson advises to define names for complicated constructs and to use these names.

I don't like the heavy braces on among others the pages 103, 104, 106, 110, 111, 116.

Much functionality of TeX has been renamed on the one hand while on the other hand some functionality has been altered under the same name. This makes an $\mathcal{AMS}$-TeX script incompatible with plain.[5] Some names? For plain's infix command `\over`, there is the prefix command `\frac`. `\align`, `\aligned`, `\alignat`, `\split`, `\multiline`, and `\gather` with their `\end...` endings, add to plain's `\eqalign`, `\eqalignno`, and `\displaylines`. Moreover, there is `\tag` superseding `\eqno`. Different names, different syntaxis but no increased functionality. All the given formulas could have been typeset by TeX with roughly the same level of *discrepancy* between the *math copy and the TeXscript*. Then we have instead of `\matrix`, and `\pmatrix`, the substitutes `\bmatrix`, `\vmatrix`, `\pmatrix`, `\Vmatrix`, and `\smallmatrix`, while the powerful and practical `\bordermatrix` disappeared from the stage, I mean is not mentioned at all. At the definition front there is `\define`, `\redefine`, and `\predefine` as substitutes for `\def` and `\let`. (No replacement of the TeXnical `\futurelet`, of course.) `\edef` etc. disappeared as such. In `\accentedsymbol` it is used from the application viewpoint. This holds for some other TeX macros (or control sequences) too: `\overfullrule=0pt`, `\cr`, `\openup`, `\noalign`, `\phantom`, `\atop` and the like, `\vbox{\hsize=...}`, `\cal`, `\dots`, `\oldstyle`, `\hoffset`, `\voffset`, `\vadjust`, and the abbreviation period.

Next some examples of the same names but different functionality. `\item` has been redefined within the `\roster` environment. I personally love plain's `\item`; happily Appendix C reassures me that it can still be used as such. In that Appendix the different use of `\footnote`, `\proclaim` and the different attitude with respect to font changes are explained as well.

2.3 Third part: Sauces and Pickles

It starts with shorthand definitions, mainly to support typing efficiency. The example on p127, is very suited within the context, but horrible from a publishing math viewpoint, and inefficient with respect to the paper used. A waste of paper, not compensated by anything![6] The final part is an alphabetic enumeration of 'everything else.' Rich in nature, but a bit out of balance with respect to the earlier attitude. The `\struts` are hidden here, although they are very useful and common in practice.

2.4 Appendices

The collection of exercises is rich. The input format of bibliographies still does not pay enough attention to abstraction of interpunction and to abstraction of the order in which the information must be supplied. Appendix C explains in detail how to use plain ànd $\mathcal{AMS}$-TeX, that is, plain commands as well as $\mathcal{AMS}$-TeX's. I guess that Appendix F about Future fonts is outdated and reality by now.

2.5 Conclusion

A wealth of material from a user point of view is provided, ready to use. I personally pity the confusion which it will bring, because no more functionality has been provided, nor is the task of typists relieved. Many more names along with modified syntaxes have been introduced. As a self-publishing author I will stay with plain and use published and reviewed macros as extensions. Not the complete superseding collections. However, I will not refrain from using $\mathcal{AMS}$-TeX, nor will I refrain from using LaTeX, if the publication at hand can be handled more effectively by those tools.

[3] Of course it is unnatural to tell a typist to type `\,`, but a `\mulspace` could have been defined with that functionality.

[4] The more so because people are used to insert blank lines and outside math mode extra blank lines are simply ignored. So, a double warning is in place.

[5] Agreed, the reader is warned for those occurrences in Appendic C.

[6] We should conform to reality: editors love to squeeze unnecessary space.

3 LAMS-TEX

This book also consists of 3 parts: Basic document preparation (≈120p.), Fancy mathematics (≈53p.), and Tables (≈84p.). The contents is somewhat unbalanced. The basics what *everybody* should know and adhere to, then the very specialized and advanced commutative diagrams for homologists or their colleagues, and finally the in depth treatment of formatting tables. The beginner, the math specialist, and the advanced table formatter. A broad audience.

3.1 Part 1: Basic Document Preparation

Perhaps inspired by the success of LATEX, LAMS-TEX starts with a section about basic document preparation. A very strong point in my opinion is the general mechanism for automatic numbering and symbolic referencing. Very strong, and I hope this can be used as a separate package. It competes with AMS-LATEX and LATEX. Apparently, the users and macro writers agree upon the need for these kinds of things. So it is relevant!

But, . . . , I missed the style options. What about 2(and more)-columns[7]? What about other sizes of the paper, a4.sty, for example? What about language flexibility? From section 6 we get that the names, such as Chapter or the similar name in another language is left to the style files. So LAMS-TEX is not enough! A bit out of the blue are the *plain* macros on p.73. A novelty, well a revival, is that footnotes start with number 1 on each page.

3.2 Part 2: Commutative Diagrams

The examples look quite complex. The language and notation look sensible. But I'm not a homologist. And leave this for whatever it is worth, keeping in mind, if ever I would need to typeset CD's, I will certainly return to this section.

3.3 Part 3: Tables

The tables part consists of modules, which can be run separately from the main part of LAMS-TEX.

3.3.1 Processing

Tables have to be TEXed separately from the main document. In the main document (floating) space has to be specified and processed. The merging can be done at the dvi-level.

The merging of the tables into the main document goes in two steps. First—in the main run—enough space is reserved to paste the table in the appropriate place. This can be done in the 'floating way.' Next at the dvi-level the tables are pasted in. Advanced, and very powerful. Hi-TEX!

With respect to the notation it is remarkable that curly braces are never used to enclose table dimensions. Is that really easier, once one has adopted the curly braces mania? I doubt it, just confusing. I was some years ago confused by LATEX's inconsistency at this point, especially with respect to the deviating conventions adopted within the picture environment, not to mention the separators of the `\item` parameter.

The section is well done, and it contains a wealth of material. It is not a surprise that a new syntax is used throughout. The table on p.201, is realistic, with a header part and 'halflines.' I pity that the 'h' is not vertically centered. A little further a table with Side SpecificationS is given, similar to the bordered matrix idea.[8]

Around p.232 row spans—called crossing rows—are introduced. The table has 3 logical columns, but the markup needs 5 columns. I mean the user deals with a 3 column table while in the mark up *he* has to think in terms of a 5 column table. Still a discrepancy between the logical mark up, and the formatting. P.234 formats complicated headers. But again, how many columns are formatted? According to the formatting 6, but logically 3, in my opinion. This remains confusing. The 3∗3-table on p.235, with a 2∗2-subtable/block in the left upper corner, is a nice challenge for table programs which claim to be able to format row spans and column spans, simultaneously. The table is symmetrical along the main diagonal, but the formatting is not symmetrical at all. It takes 9 rows and 3 columns!

At the end notes and footnotes to tables are treated. Moreover, the rules can vary in length and thickness. Very powerful!

What I missed is typesetting of simple tables with hardly no mark up information provided by the user. No need for a preamble, and no need for &-s and `\cr`-s; just spaces and carriage returns as separators. Similar to Cowan (1985).

3.4 Conclusion

LAMS-TEX is certainly up-to-date with some very powerful features. It combines basic mechanisms like general automatic numbering and symbolic referencing with advanced and esoteric commutative diagrams. A bit out of balance. The table macros are very powerful, but not simple to use. I missed the class of simple tables, and the tables which extend the page.[9] Both do occur in scientific publishing.

4 Some afterthoughts

How come that at the time of the birth of the lxiii project, 1989, LAMS-TEX appeared, and that there is no cooperation? The weaknesses of LATEX, which LAMS-TEX has tried to overcome—which are also the targets of lxiii—will be reprogrammed from scratch. It sounds

[7] Agreed, math papers generally don't take 2 or more columns.

[8] The SGML community talks about row stubs.

[9] Mentioning portait and landscape as variants to be handled at the driver level would have been nice.

like a gigantic waste of energy. And this is not the only effort in that direction. Cooperation remains apparently difficult. Perhaps we should be more modest in the spirit of Rogers[10]

> 'It's also unfortunate that Dr Spivak, as well as many others, choose to embed macros of this nature in large packages such as LAMS-TEX, AMS-TEX, LATEX, etc. I would rather see them made available as self-contained modules that can be easily incorporated into macro packages designed to accomplish specific purposes.'

I completely agree with that attitude.

References

[1] Cowan, R.F. (1985): Tables.tex (from the file server).

[2] Spivak, M.D. (1986): The Joy of TEX, AMS, ISBN 0-8218-2999-8 (second printing).

[3] Spivak, M, D. (1989): LAMS-TEX The Synthesis, TEXplorators. 3701 W. Alabama, Suite 450–273, Houston, TX 77027.

[4] Swanson, E. (1986): Mathematics into type, AMS, ISBN 0-8218-0053-1. (reprinted).

[10] Quoted from On Contrarian Views, TEXline 14.

Table of Contents TUGboat

Volume 12.3, 12.4 and 13.1

December 1991 / April 1992

TUGboat tables of contents files are on `math.utah.edu` in `f̃tp/pub/tex/pub/tugboat`, also accessible via `tuglib@math.utah.edu` server by 'send index from tex/pub/tugboat'.

1 TUGboat 12.3 (December 1991)[1]

- **Nelson Beebe**
  President's introduction, p. 351–352
- **Nico A.F.M. Poppelier**
  Two sides of the fence, p. 353–358
- **Laurie J. Petrycki**
  Comparing TeX and traditional typesetting for the composition of a textbook, p. 359–366
- **Frederick H. Bartlett**
  Contra-LaTeX, or what really works in the publishing world, p. 367–371
- **Howard Ratner and Kenneth Dreyhaupt**
  TeX in a book production department, p. 372–376
- **Berthold K.P. Horn**
  `dvi` *and eps: The ideal author-to-publisher interface?,* p. 377–381
- **Neil A. Weiss**
  Producing a book using TeX: How the process works, p. 382–386
- **Samuel E. Rhoads**
  Authors new to TeX publish a textbook with a publisher new to TeX, p. 387–392
- **Colleen Brosnan**
  The "Five Cs": A guide to successful publication using TeX, p. 393–396
- **Anita Z. Hoover**
  LaTeX/TeX user: A typist, or typesetter?, p. 397–400
- **John Lavagnino**
  Simultaneous electronic and paper publication, p. 401–405
- **Robert W. McGaffey**
  SGML versus/and TeX, p. 406–408
- **Andrew E. Dobrowolski**
  Typesetting SGML documents using TeX, p. 409–414
- **C. M. Sperberg-McQueen**
  Specifying document structure: Differences in LaTeX and TEI markup, p. 415–421
- **Takashi Kakiuchi, Yuki Kusumi, Yoshiyuki Miyabe, and Kazu Tsuga**
  A structured document preparation system—Autolayouter version 2.0—an enhancement for handling multiple document types, p. 422–429
- **Williams**
  Refining a process, p. 430–433
- **Semenzato and Edward Wang**
  A text processing language should be first a programming language, p. 434–441
- **Michael Vulis**
  Should TeX be extended?, p. 442–447

- TUG 1992 annual meeting, Portland, Oregon, p. 448
- Institutional members, p. 449–450
- TUG membership applications, p. 451–452
- TeX consultants, p. 453–453
- Index of advertisers, p. 468–468

2 TUGboat 12.4 (December 1991)[2]

- **Cay S. Horstmann**
  Automatic conversion from a scientific word processor to TeX, p. 471–478
- **Dennis S. Arnon and Sandra A. Mamrak**
  On the logical structure of mathematical notation, p. 479–484
- **Kees van der Laan**
  Math into BLUes, p. 485–501
- **Michael J. Downes**
  Dialog with TeX, p. 502–509
- **Jackie Damrau and Michael Wester**
  Form letters with 3-across labels capability, p. 510–516
- **Mark A. Roth**
  Typesetting forms with LaTeX, p. 517–522

[1] 1991 TUG Conference Proceedings — Part 1.

[2] 1991 TUG Conference Proceedings — Part 2.

- **Peter Flynn**
  Developing a pop-up help facility for TEX on PCs, p. 523–527
- **Malcolm Clark, Brian Hamilton Kelly, and Niel Kempson**
  7 bits good, 8 bits bad or 'The eight-bit blight', p. 528–533
- **Friedhelm Sowa**
  Bitmaps and halftones with BM2FONT, p. 534–538
- **Yannis Haralambous**
  TEX and those other languages, p. 539–548
- **Don Hosek**
  Siamese TEX: Joining dvi files at the hip and other novel applications of vf files, p. 549–553
- **Alan Hoenig**
  When TEX and METAFONT *talk: Typesetting on curved paths and other special effects*, p. 554–557

- Participants at TUG'91, p. 558–561
- Calendar, p. 562–563
- The Donald E. Knuth Scholarship for 1992, p. 563–564
- GUTenberg'92, Les Diablarets, Switzerland, p. 564–565
- TUG 1992 annual meeting, Portland, Oregon, p. 570
- Index of advertisers, p. 588

3 TUGboat 13.1 (April 1992)

- **Malcolm Clark**
  Prez says, p. 5
- **Nelson H. F. Beebe**
  President's introduction, p. 6–10
- **Barbara Beeton**
  Editorial comments, p. 10–11
- Samuel B. Whidden, 1930–1991, p. 11–12
- **Graham Asher**
  Inside Type & Set, p. 13–22
- **Claudio Beccari**
  Computer Aided Hyphenation for Italian and Modern Latin, p. 23–33
- **Sebastian Rahtz**
  Invisibility using virtual fonts, p. 34–36
- **Toby Thain**
  Packing METAFONT*s into PostScript*, p. 36–38
- **C. Mylonas and R. Whitney**
  Modern Greek with adjunct fonts, p. 39–50
- **Frank Mittelbach**
  Comments on 'Filenames for Fonts' (TUGboat 11#4), p. 51–53
- **TUG DVI Driver Standards Committee**
  DVI driver standard, level 0, p. 54–57
- **Victor Eijkhout**
  New books on TEX, p. 57–58
- **Nico Poppelier**
  New books on LATEX, p. 58–59
- **Alan Hoenig**
  Just plain Q&A: Of partitioned matrices and doublespacing, p. 60–62
- **L. Siebenmann**
  Elementary text processing and parsing in TEX — the appreciation of tokens —, p. 62–73
- **Victor Eijkhout**
  The bag of tricks, p. 74
- **Victor Eijkhout**
  Erratum: Oral TEX, TUGboat 1, no. 2(2), pp. 272–276, p. 75
- **Jonathan Fine**
  Some basic control macros for TEX, p. 75–83
- **Victor Eijkhout and Ron Sommeling**
  Self-replicating macros, p. 84
- **Piet Tutelaers**
  A font and a style for typesetting chess using LATEX or TEX, p. 85–90
- **Kees van der Laan**
  Tower of Hanoi, revisited, p. 91–94
- **Jackie Damrau**
  The LATEX column, p. 94–95
- **Harold Thimbleby**
  Erratum: 'See also' indexing with Makeindex, TUGboat 1, no. 2(2), p. 290, p. 95
- **Frank Mittelbach and Chris Rowley**
  LATEX 2.09 ↪ LATEX3, p. 96–101

- Cahiers GUTenberg #9 and #10–11, p. 101–105
- Calendar, p. 106–107
- EuroTEX 92, Prague, 14–18 September 1992, p. 107–108
- Barbara Beeton
  Production notes, p. 108–109
- Coming next issue, p. 109
- Institutional members, p. 110–111
- TUG membership application, p. 112
- Index of advertisers, p. 126
- TEX consulting and production services, p. 127

Tijdschriften zusterverenigingen

Het secretariaat ontvangt van de zusterverenigingen ondermeer de volgende tijdschriften:

1. Cahiers GUTenberg (TeX Usersgroup Frankrijk)
2. Die TeXnische Komödie (TeX Usersgroup Duitsland)
3. TeX bulletin (TeX Usersgroup Tsjecho Slowakije)
4. SGML-Bulletin Holland (SGML Usersgroup Holland)

Deze tijdschriften zijn in te zien bij de tweejaarlijkse NTG bijeenkomsten. Geïnteresseerd in bepaalde artikelen of bepaalde tijdschriften? Neem dan even contact op met het secretariaat (bij voorkeur tijdens de bijeenkomsten).

Cahiers GUTenberg #9 (Jul 1991)

- **J. André**
  Éditorial: un nouveau style pour les Cahiers GUTenberg, p. 1–31
- **Ph. Louarn**
  Lucida: une fonte complète pour LaTeX et son installation, p. 32–40
- **O. Nicole**
  trad.- The Economist polit ses polices, p. 41–48
- **V. Quint** *et al*
  Grif et l'édition de documents structurés, p. 49–65
- **H. Thomas**
  Typographie du jeu d'échecs, p. 66–74
- **A. Heck**
  StarTeX p. 75–78
- **Y. Haralambous**
  Quand TeX rencontre Mozart, p. 79–82
- **B. Gaulle**
  L'association. . . fait la force, p. 83–85
- **É. Picheral**
  Distribution MlTEX v.3.14 pour Sun, p. 86–87

Cahiers GUTenberg #10,11 (Sep 1991)

- **B. Malyshev, A. Samarin & D. Vulis**
  Russian TeX, p. 1–6
- **T. Jurriens**
  TeXniques in Siberia, p. 7–14
- **J. Knappen**
  TeX and Africa, p. '15–24
- **O. Boughaba, S. Boutalbi, & M. Fanton**
  Vers une version arabisée de TeX, p. 25–44
- **B. Malyshev & A. Samarin**
  TeX Integrated Shell for IBM PC, p. 45–56
- **J. Zlatuška**
  Automatic generation of virtual fonts with accented letters for TeX, p. 57–68
- **Y. Haralambous**
  ScholarTeX, p. 69–70
- **J. Braams**
  Babel, a multilingual style-option system, p. 71–72
- **M. Fanton**
  TeX: les limites du multilinguisme, p. 73–80
- **J. Schrod**
  An International Version of MakeIndex, p. 81–90
- **P. Bacsich, E. Heyes, P. Lefrere & G. Yarwood**
  Conversion of Microsoft word into LaTeX, p. 91–92
- **M. Lavaud**
  AsTeX: an integrated and customizable multiwindow envirronment for scientific research, p. 93–116
- **Larsen & A.F. Jensen**
  Tailored database publishing with TeX, p. 117–134
- **B. Leguy**
  Drawing tree structures with GWEZ, p. 135–146
- **K. van der Laan**
  Math into BLUes : Sing your song, p. 147–170
- **A. Binding**
  Organizing a large collection of stylefiles, p. 171–184
- **A.E. Dobrowolski**
  Typesetting SGML Documents Using TeX, p. 185–196
- **Ch. Cérin**
  Vers la construction de macros de mise en coleur pour TeX, p.'197–208
- **M. Laugier**
  Composition des formules chimiques en TeX, p. 209–221

Cahiers GUTenberg #12 (Dec 1991)

- **B. Gaulle**
  Éditorial: á propos d'erratum, p. 1–2
- **E. Göpelt & B. Schmid**
  WYSIWYG-TEX-editors on the basis of object-oriented system technology, p. 3–12
- **M. Spivak**
  LAMS-TeX: A Public Domain Document Preparation System Extended AMS-TeX, p. 13–20
- **P.A. MacKay**
  Un regard sur les pixels. Obtention de fontes de qualité pour imprimantes à lser à 300 dpi grâce à METAFONT, p. 21–36
- **M. Goossens & E. van Herwijnen**
  Introduction à SGML, DSSSL et SPDL, p. 37–56
- **J. André & Ph. Louarn**
  Notes en bas de pages: comment les faire en LaTeX?,

p. 57–70

- **A. Samarin & A. Urvantsev**
  CyrTUG, le monde TEX en cyrillique, p. 71–74
- **H. Kołodziejska**
  TEX en Pologne p. 75–77
- **J. André**
  Lu, vu ou entendu, p. 78–83

- Table des matières pour l'année 1991, p. 84–85

Die TEXnische Komödie #3.1 (Mai 1991)

- **Luzia Dietsche & Joachim Lammarsch**
  Protokoll der 4. Mitgliederversammlung von DANTE, p. 4–11
- **Erich Neuwirth und Walter Obermiller**
  Bericht über die 10. Tagung der deutschsprachigen TEX Interessenten, p. 13–16
- **Thomas Esken**
  Eine Bedienoberfläche fur das emTEX-Systems, p. 17–20
- **Konrad Bernlöhr**
  VIEWDVI — Ein leistungsfähiger Previewer für Personal computer, p. 20–23
- **Oliver Schurr**
  Grafikeinbindung (!) in TEX mit HPTOMF. Ein Erfahrungsbericht, p. 23–24
- **Lothar Meyer-Lerbs**
  Rezensionen: Zwei Klassiker der Typographie, p. 25–26
- **Lothar Meyer-Lerbs**
  Tips zum Tabellensatz, p. 27–29

Die TEXnische Komödie #3.2 (Aug 1991)

- **Joachim Lammarsch**
  Neues von DANTE, p. 4–7
- **Joachim Lammarsch**
  Wahlen zum Board of Directors, p. 7–8
- **Joachim Lammarsch**
  Rücktritt des European Coordinators , p. 9
- **Friedhelm Sowa**
  Grafikintegratrion mit BM2FONT, p. 10–14
- **Heinz Werntges**
  hp2xx — Ein Konverter für HPGL-Daten, p. 14–20
- **Torsten-Karl Strempel**
  LATEX — Programm zum Briefeschreiben, p. 21–25
- **Rainer Schöpf**
  Softwareverteilung durch elektronische Netze I — Die TEX-Server in Heidelberg und Stuttgart, p. 26–32
- **Joachim Lammarsch**
  QMS PS 815 Laserdrucker), p. 32–33
- **Luzia Dietsche**
  . . . der hat was zu erzählen, p. 33–34
- **Peter Schmitt**
  Einige Bemerkungen zu den DC-Fonts, p. 35–37

Die TEXnische Komödie #3.3 (Okt 1991)

- **Luzia Dietsche & Joachim Lammarsch**
  Protokoll der 5. Mitgliederversammlung von DANTE, p. 4–16
- **Joachim Lammarsch**
  Ein Resümee, p. 17–18
- **Andreas Dafferner & Rainer Schöpf**
  Ein Aufenthalt in Elbflorenz, p. 19–21
- **H.W. Kisker**
  RUMgraph — Einbindung von Pixel-Bildern in ein TEX-Dokument, Teil 1, p. 22–33
- **Hartmut Wiechern**
  Test zweier Atari-Versionen, p. 34–37
- **Michael Baas**
  Briefe mit LATEX — Ein Überblick, p. 37–40

EuroTEX 92 announcement

Jiri Vesely

jvesely@cspguk11

14–18 september 1992

Dear TEXfriends,

EuroTEX 92 is organized by CSTUG in collaboration with Charles University and Czech Technical University under the auspices of both Rectors. It takes place in Prague on:

14 September 1992 — 18 September 1992

(Please, make a note in your diary.) As standard accommodation we offer a relatively modern student hostel Kajetanka. We plan to organize for every morning the transfer by bus to Czech Technical University where the programme will be held (it is within some 25 minutes walk from Kajetanka). Lunch and dinner will be served at the conference site.

We intend to arrange an opening party on Monday evening, a concert in a historical hall typical of Prague on 16 September 1992, and some other events. To those who are coming during Monday morning we would like to offer a sightseeing tour on Monday afternoon. To keep things within everybody's budget, we offer the main programme as a package: whole programme from Monday to Friday morning (accommodation in double rooms, full pension from Tuesday to Thursday, opening party on Monday evening, concert, breakfast on Friday, conference fee and proceedings) for about 300 DM. Those who would prefer a single room should pay extra 60 DM. (Payment details, account numbers, etc., will be sent to you later.) For accompanying persons, a special programme will be organized including visits to galleries, places of interest, etc. For *additional* 50 DM a day, a limited number of participants may stay on till Sunday (one or two days more) either for tutorials or just to enjoy meeting friends and good beer in some of the pubs Good Soldier Svejk would visit.

Currency exchange rates: 1 DM is about 18 CSK (Czechoslovak krown), USD 1 is about 28 CSK
You can exchange within Czechoslovakia without any problem/obligation, no visa is required for most countries (all of Europe and Nothern America)

Why come to EuroTEX 92 ? Except for invited talks of leading specialists you will have a chance to listen lectures on different TEX applications and meet TEX friends from many countries. The meeting is the first offering really extensive contacts with people from former East European countries ("from behind the iron curtain" – can you still remember that?). It takes place in the Golden Heart of Europe – Prague, one of the nicest capitals in Europe. You can visit it for a surprisingly low price: since we would like to make EuroTEX 92 in Prague accessible to the majority of TEXfans from all over Europe, we arrange it on a modest but good level.

We would like to encourage people to submit papers for EuroTEX 92. Suggested topics of special interest include the following themes:

- Quo vadis, TEX?
- National versions and standardization
- Non-standard applications
- The use of TEX for small/newly emerging enterprises

The above list is by no means complete, and interesting contributions relevant to TEX are sought.

Important dates for submitting paper proposals:

- 1 May 1992: two-page (max.) abstract to be sent to the address of the Programme Committee chairman:

  Jiri Zlatuska
  Masaryk University
  Buresova 20
  601 77 Brno
  Czechoslovakia
  e-mail: zlatuska@cspuni12.bitnet
- 1 June 1992: acceptance/rejection announcement to the author
- 10 July camera-ready papers due to arrive to the Programme Committee chairman

EuroTEX 92 Programme Committee:

Peter Abbott	Jacques Andre
Jana Chlebikova	Bernard Gaulle
Karel Horak	Joachim Lammarsche
Erich Neuwirth	Petr Novak
Stefan Porubsky	Phillip Taylor
Jiri Vesely	Jiri Zlatuska

Those who are interested in taking part are kindly asked to fill in the following form and send it via e-mail to the address:

eurotex at cspguk11.bitnet

```
I intend to take part in EuroTeX 92, Prague (14.-18.9.1992)

name:
userid:
node:

Please send  a more detailed  information on EuroTeX  to the
following e-mail address (please, fill in):

-------------------------------------------------------------

(or: print the following form, fill in using block letters,
and send to the address:

                CS TeX --- EUROTEX 92
                Mathematical Institute
                Sokolovska 83
                186 00 PRAHA 8 - Karlin
                Czechoslovakia

   *************************************************
   *                                               *
   *  Name: .....................................  *
   *  Institution: ..............................  *
   *  Position: .................................  *
   *  Address (business or home - circle one):     *
   *  ...........................................  *
   *  ...........................................  *
   *  ...........................................  *
   *  ...........................................  *
   *                                               *
   *    I intend to visit EuroTeX 92 in Prague     *
   *       (September 14. -- 18.(20.),1992)        *
   *                                               *
   *  ...........................................  *
   *                 (Signature)                   *
   *                                               *
   *************************************************

On behalf of organizers:

Karel Horak            Jiri Vesely           Jiri Zlatuska
```

NTG's Advanced TEX course
Insights & Hindsights

David Salomon

Groningen, 15–19th June, 1992

This advanced course, with no hands-on, is aimed at those TEX and/or LATEX users who are ready for a deeper insight into the TEXnigma. An extra one-day introductory course can be organized for inexperienced users, if there is enough demand.

Course outline

- **Day 1:**
  Introduction to TEX. The cm fonts. Tables (examples). Math typesetting (some advanced features). Modes of TEX.
- **Day 2:**
  Boxes & glue. Paragraphs & Horizontal mode.
- **Day 3:**
  Macros (advanced features & examples). Leaders. Tokens & \toks registers.
- **Day 4:**
  File I/O in TEX. Examples of two-pass jobs. Output routines.
- **Day 5:**
  Insertions. The line break algorithm (in detail). The page break algorithm (in general).
  The last afternoon will be spent answering specific questions, and discussing specific topics proposed by the participants.

Each day is a separate module, so it is possible to benefit from selected parts of the course.[1]

Whom? When? Where? How much?

- **For whom?**
  TEX or LATEX users with some hands-on experience will benefit most.
- **When?**
  June, 15–19th, 1992, 9.00–12.00 & 14.00–17.00hr.
- **Where?**
  RUG Paddepoel, Groningen, The Netherlands, Zernike Gebouw, room ZG114 and ZG107.[2]
- **How much?**
  For NTG members and members of similar user groups a flat Fl 100,– (Yes, only a hundred, but no lunches and refreshments are included; the University 'Mensa' is quite cheap and restaurants close by). For non-NTG members Fl 500,–.[3]

The teacher is David Salomon, an experienced TUG instructor, well known for his lucid tutorials in TUGboat.

How to subscribe?

Via snail

Send a note, mentioning 'Insights in TEX'-course along with name, complete address, phone number and email, to:

Kees van der Laan,
Hunzeweg 57,
9893 PB, Garnwerd,
The Netherlands
(+31 5941–1525)

Mentioning of topics to be treated in the last afternoon is very much appreciated.

Via email

Send a subscription note, mentioning 'Insights in TEX'-course along with name, complete address, phone number (and email), to:

cgl@rug.nl.

Payment

As soon as possible to:

Penningmeester NTG,
Giro: 1306238
mentioning 'Insights in TEX' course.

[1] At the same flat fee for members, however. For others—why not become a member?—Fl 100,– for each day, with at minimum Fl 250,–.

[2] For lodging contact VVV, Naberpassage 3, 9712 JV, Groningen, The Netherlands. Phone: +31 50 139700, Fax: +31 50 136358.

[3] For those who don't believe in a quality course at that fee — courseware included —, it should be mentioned that NTG subsidizes the project, so it is a unique opportunity. NTG's philosophy is that TEX education is paramount for people interested in high-quality typesetted documents.

Some Good Reasons

for subscribing to

Insights & Hindsights

You are a LATEX user, and

- attracted by the magic
- like to work more effectively
- like to learn more about TEX
- need to write (small) macros
- like to learn from your friends

You are a LATEX macro writer, and

- like to understand the basics
- like to write some real jewels
- like to write in TEX

You are a TEX user, and

- like more of it
- like to drop your questions
- like to publish in TUGboat

You are a (TEX) hacker:

- watch the teacher
- improve the notes
- be the next teacher

You are a novice?

Come along and meet the people

Groningen is a nice place, anyway

TEXies are good company, join them

And everybody likes to get the most out of it!

Moreover, it is a unique opportunity.
NTG won't organize such a course in the near future.
David's notes will appear in the MAPS; a continuing story.