MINUTES

M&APS

APPENDICES

Nederlandstalige TEX Gebruikersgroep (NTG)

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

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

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

De NTG biedt haar leden onder meer het volgende:

- Tweemaal per jaar een NTG-bijeenkomst.
- Tweemaal per jaar het uitgebreide NTG tijdschrift: MAPS (Minutes and APpendiceS).
- Speciale MAPS uitgaven (o.a. TEX-cursus, FAQ, en PR-set).
- De *dubbele* 4allTEX CD-ROM met een volledige en direct te gebruiken TEX DOS/Windows/OS-2/Linux implementatie inclusief een uitgebreide verzameling van utilities. De CD-ROM bevat zeer veel documentatie, inclusief discussielijsten van vele jaren, *alle* MAPS uitgaven, en zeer veel tutorials.
- Korting op (buitenlandse) TEX congressen en cursussen en op het lidmaatschap van TUG.
- Jaarlijks een ledenlijst met informatie over welke software/hardware, in relatie met TEX, wordt gebruikt.
- De discussielijst (listserver) TEX-NL waarop vragen worden gesteld en ervaringen uitgewisseld.
- De fileserver TEX-NL waarop algemeen te gebruiken 'TEX-produkten' staan.
- Het NTG FGBBS Bulletin Board met ruim 1 GByte aan TEX en aanverwante software.
- Aktiviteiten in werkgroepen. Enkele belangrijke werkgroepen zijn: 'Nederlandse TEX', 'PC's en TEX', 'educatie' (cursussen), en 'communicatie'.

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

De contributie voor een persoonlijk lidmaatschap bedraagt *f* 90,–, de contributie voor een instituutslidmaatschap bedraagt *f* 245,–. Een instituutslidmaatschap geeft het recht om drie personen aan te wijzen die informatie welke aan de leden wordt verstuurd, ontvangen. Van die drie personen dient één persoon te worden aangewezen als rechtsgeldige vertegenwoordiger van het bedrijf/instituut, een ander als vervangend vertegenwoordiger. Indien meer leden per bedrijf/instituut lid willen worden, geldt als additioneel tarief *f* 65,– per persoon. Voor studenten geldt een tarief van *f* 60,– (geen stemrecht; bewijs van inschrijving vereist). Voor afwijkende regelingen dient contact met het bestuur opgenomen te worden. Een gecombineerd NTG/TUG lidmaatschap voor 1997 bedraagt *f* 167,50 per jaar (i.p.v. *f* 90,- + $ 55).
Belgische leden kunnen de lidmaatschapskosten van BF 1660 (individueel), BF 4520 (instituutslidmaatschap) of BF 3090 (NTG/TUG lidmaatschap) overmaken op de NTG Belgische postgiro te Veldegem (zie hierboven).

Nederlandstalige TEX Gebruikersgroep

Aanwezig : W. van Beers (Hogeschool 's-Hertogenbosch); P.G.M. Bezembinder (Thieme); P. Bloemen (K.M.); B. de Boer; J. Braams; R. Breedveld; L. de Coninck (de Kraal); G. van den Dobbelsteen (LOGIN BV); W. Dol; W.E. Dolman (Hogeschool van Amsterdam); G. Draisma (Erasmus Universiteit); E. van Eynde (K.U. Leuven); C.M. Fortuin (Hogeschool Gelderland); E. Frambach (RUG); B.J. Geels (Adriaan Roland Holst school); G. Geens (TFCG/IMEC); F. Goddijn (FGBBS); W.J. van de Guchte; J. Hagen (Pragma); T.J. Harkema (Friesland College); H. Haverkort; J. Hellings (UvA); R. van der Heijden (Hogeschool Utrecht); A. Heijs (Staring Centrum); T. Hoekwater (Wolters Kluwer Academic Publishers); T.A. Jurriens (Rijksuniversiteit Groningen); C.H.A. Keyer (Hogeschool van Amsterdam); A.H. Kotterink; N.S. Kroonenberg (RUG); C.G. van der Laan; F.J. Lauwers; A.W.J. van der Meer (UT); F.D. Mesman (Elsevier Science Publishers); C.C.M. Moes (TUD); A. Molenaar (TUD); G.J.H. van Nes (ECN); J.H.B. Nijhof (RUG); T. Otten (Pragma); S.A.M. Pepping (Elsevier Science Publishers); R. Rietman; R.B.J. Pijlgroms (Hogeschool van Amsterdam); R. Smedinga (RUG); W. Smit; A. Soos (UT); A. Steegstra (Vrije Universiteit); E. Steuten (Hogeschool van Amsterdam); E. van der Storm; P. Tutelaers (TUE); E. Ulijn (TUD); G. Vlak; J. van Walen (Hanzehogeschool); J.E. van Weerden (UU); J.J. Winnink;

Notulist : Frans Goddijn

1 Opening

Voorzitter Erik Frambach opent de zeventiende NTG-bijeenkomst en dankt de gastheer Kees Keizer.

2 Notulen van de zestiende bijeenkomst van 5 september 1995

De notulen worden ongewijzigd vastgesteld.

Naar aanleiding van het verslag vraagt Jos Winnink of de najaarsvergadering van 24 oktober nog is te verplaatsen. Misschien is het tijdstip niet ideaal.
Jules van Weerden zal de mogelijkheden daartoe bekijken.

3 Ingekomen stukken en mededelingen

Secretaris Gerard van Nes geeft een enthousiaste uiteenzetting over de binnengekomen tijdschriften van diverse zusterverenigingen. Ook geeft hij een overzicht van de stand van zaken betreffende de binnengekomen bestellingen voor de 4TEX CD.

4 NTG ledenvergadering

4.1 Jaarverslag secretaris

Het jaarverslag wordt ongewijzigd vastgesteld.

4.2 Jaarverslag penningmeester

Kees van der Laan vraagt of de verschillende balansdata, 31 december en 1 februari correct zijn. Wietse Dol licht toe dat dit inderdaad in orde is. Vorig jaar was er enige uitloop in verband met de contributiebetalingen.

Voorts, zo licht Wietse Dol toe, is er een kleine schommeling in de balans ten gevolge van de CD-omzet. Deze lijkt bovendien groter te zijn door de opening, tijdens de rit, van een speciale CD-rekening.

Kees van der Laan vraagt zich af hoe het komt dat EuroTEX buiten de NTG-boekhouding is gebleven. Wietse Dol antwoordt hierop dat EuroTEX ook nooit op de NTG-begroting heeft gestaan. Het verslag van de penningmeester dat nu aan de orde is, betreft *deze* begroting, niet die van EuroTEX dat niet door de penningmeester of onder diens verantwoordelijkheid is georganiseerd.

Erik Frambach vertelt dat een belangrijk leerpunt van de ervaring met EuroTEX is, dat bij dergelijke projekten een veel meer gestructureerde aanpak nodig is. Echter, als deze manier van werken een voor-vereiste bij EuroTEX zou zijn geweest, was het daardoor gezien de beperkte mankracht ook onmogelijk geweest EuroTEX te laten plaatsvinden.

Simon Pepping informeert naar de status van het CD-projekt, waarop Erik Frambach vertelt dat dit aan de orde zal komen bij zijn lezing over het 4allTEX projekt.

Jules van Weerden vraagt of belangstellende leden via email een financieel overzicht van EuroTEX kunnen ontvangen. Daarvoor wordt gezorgd.

Editors van deze MAPS zijn: Gilbert van den Dobbelsteen, Wietse Dol, Gerard van Nes, en Jos Winnink. Oplage MAPS: 350.

Simon Pepping zou willen zien dat er formaliteiten worden geregeld voor het verdere verloop van het 4allTEXprojekt. Erik Frambach licht toe dat het 4TEX team ijn Bonn is geweest, waar is overlegd met Addison Wesley. Deze uitgeverij is van plan de productie en de financieele afhandeling op zich te gaan nemen. Kees van der Laan zou graag zien dat NTG als belangrijke factor in het beginstadium van 4TEX betrokken blijft bij het projekt en Erik Frambach zegt toe dat dit in enige vorm ook zal kunnen.

4.3 Royementen

De heer Meinema heeft alsnog betaald, de rest komt in aanmerking voor royement en de vergadering gaat hiermee akkoord. Geroyeerd zijn: G.E.J. Bulder, A.L. Hoffman, H. Kelderman, F. van Manen en L. Vangilbergen

5 Verslag commissie voor kascontrole en vaststelling nieuwe commissie voor kascontrole

Simon Pepping meldt namens de commissie (Phons Bloemen en hijzelf) dat de boekhouding in orde is bevonden en dat de penningmeester decharge wordt verleend.

Het verslag van de commissie wordt aanvaard en de nieuwe kascontrolecommissie zal bestaan uit Phons Bloemen en Kees van der Laan.

6 Bestuursverkiezingen

Wietse Dol en Frans Goddijn treden reglementair af en Johannes Braams treedt af wegens tijdgebrek.

Wietse Dol en Frans Goddijn worden herkozen.

Als nieuwe bestuursleden worden Taco Hoekwater en Hans Hagen voorgedragen.

Erik Frambach en Gerard van Nes blijven als bestuursleden aan.

Erik Frambach leverde een uitgebreide bestuurstaakomschrijving en de conclusie was dat de bestuursomvang te klein was.

Jules van Weerden vindt het een bezwaar dat deze bestuursuitbreiding onaangekondigd was. Kees van der Laan drukt het bestuur op het hart geen *waterhoofd* te worden. Wie enige activiteit ontplooit, behoeft daarom niet meteen in het bestuur plaats te gaan nemen. Johannes Braams licht toe dat we statutair nog steeds binnen het maximum blijven en dat er domweg teveel taken blijven liggen in het kleinere bestuur.

Frans Goddijn stelt voor dat hij uit het bestuur stapt om plaats te maken voor Hans Hagen. Hiermee zal er niets veranderen in de activiteiten en toch zou er zo aan het bezwaar van Kees van der Laan tegemoet worden gekomen. Kees van der Laan zou deze optie graag in stemming willen brengen maar op aanwijzing van Johannes Braams gaat het voorstel dat het verst gaat eerst in stemming.

Als nieuwe bestuursleden worden Taco Hoekwater en Hans Hagen gekozen met 2 tegenstemmen en 6 onthoudingen.

Het aftredende bestuurslid Johannes Braams wordt door Erik Frambach voorgedragen voor het erelidmaatschap vanwege diens verdiensten (BABEL!) voor de gehele TEX-gemeenschap. Dit voorstel wordt door de vergadering aangenomen en Erik Frambach biedt Johannes Braams een Star Trek gezelschapsspel aan voor het gezin Braams.

7 Miscellaneous

7.1 MAPS

Wietse Dol is MAPS-hoofdredacteur geworden.

De MAPS Award, traditioneel een puntzak met strechable spaces (spekkies) zal gaan naar Christiena Thiele voor haar liefdevolle arbeid aan de uitwerking van het grote Knuth-gesprek.

Ook gaat er een MAPS Award naar Kees van der Laan voor diens gehele oeuvre, dat nog steeds in aanbouw is, maar met name voor de nieuwe inspanningen op het gebied van Metafont & Metapost.

7.2 TEX-NL

De NTG zal via Jules van Weerden de jaarlijkse kosten voor het NTG domain betalen.

7.3 4TEX CD

De informatie hierover is eerder in de vergadering gegeven.

7.4 NTG WWW

In TUGboat zou het nieuwe adres moeten komen te staan.

7.5 WWW/ FTP server

Er is door NTG/ 4TEX een subsidie gegeven aan deze server in Utrecht voor de aanschaf van een 4Gigabyte harde schijf waarop een kopie van het CTAN zal komen te staan, plus alle NTG WWW zaken.

7.6 FGBBS

Frans Goddijn heeft schriftelijk verslag gedaan van de ontwikkelingen.

7.7 TUG-LUG

Kees van der Laan merkt op dat in de TUGboats geen melding wordt gemakt van NTG-activiteiten. Erik Frambach vertelt dat telkens weer alle informatie wordt doorgegeven, maar dat daar kennelijk niets mee wordt gedaan. Wietse Dol concludeert dat TUG steeds meer moeite heeft om aan de servicevraag te voldoen en hij voorziet dat dit probleem alleen maar zal groeien.

7.8 LUG / UNIX CD

Er is een complete TEX-installatie voor 35 UNIX systemen beschikbaar gekomen.

7.9 Werkgroepen

In het vorige verslag stond: “Johannes Braams zal de diverse werkgroepen aanspreken en inventariseren welke wegens inactiviteit kunnen worden opgeheven.”

In de volgende MAPS zal een lijst met nog actieve werkgroepen worden opgenomen.

Kees van der Laan verklaart actiever te willen worden in de werkgroep Edukatie en hij zal hiertoe het initiatief nemen.

Johannes Braams meldt dat werkgroep 13 voor Nederlandse afbreekpatronen ook in stand moet worden gehouden.

8 Rondvraag en sluiting

Het thema van de volgende bijeenkomst zal “Graphics en PostScript” zijn. Kees van der Laan organiseert een tutorial.

Gerard van Nes meldt dat we in juni 1997 terecht kunnen bij de TU in Delft. We moeten nu reeds polsen waar we in het najaar van 1997 en het voorjaar van 1998 te gast zouden kunnen zijn.

Erik Frambach zou graag ideeeen ontvangen van mensen die plannen kunnen bedenken voor activiteiten ter gelegenheid van het tienjarig bestaan van de NTG over 2 jaar. De Poolse LUG viert dan haar eerste lustrum. Aanleiding om iets samen te doen? Gea Vlak merkt op dat er ook tijdig een begroting voor dient te worden gemaakt.

Voorzitter Erik Frambach sluit de vergadering en hoopt iedereen weer te begroeten op de volgende bijeenkomst, 24 oktober, in Utrecht.

1 TEX kalender 1996/1997

24 okt '96	NTG (18 e)	Utrecht
voorjaar '97	NTG (19e)	Delft
najaar '97	euroTEX'97	Barcelona, Spanje

2 Glossary

Gebruikersgroepen

TUG	:	TEX Users Group
LUG	:	Local Users Group
CSTUG	:	LUG Tsjecho Slowakije
CyrTUG	:	LUG USSR
DANTE	:	LUG Duitsland
GUTenberg	:	LUG Frankrijk
HunTUG	:	LUG Hongarije
ITALIC	:	LUG Ierland
JTUG	:	LUG Japan
Nordic	:	LUG Scandinavië, Denemarken, en IJsland
NTG	:	LUG Nederland en Belgie
SibTUG	:	LUG Siberië
UKTUG	:	LUG Engeland
YUNUS	:	LUG Turkije
GUST	:	LUG Polen

Bulletins/journals

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

Diversen

AMS	:	American Mathematical Society
BoD	:	Board of Directors
SGML	:	Standard Generalized Markup Language
ltxiii	:	LATEX 3.0
FGBBS	:	NTG's Bulletin Board

3 NTG's TEX Bulletin Board Systeem

Op het TEX Bulletin Board van de Nederlandstalige TEX Gebruikersgroep (FGBBS) is een zo volledig en actueel mogelijke TEX, emTEX, LATEX, TEX-NL en MusicTEX collectie beschikbaar voor alle bezitters van een modem. Het BBS is kosteloos toegankelijk voor iedereen en er zijn geen beperkingen aan de hoeveelheid bestanden die kunnen worden opgevraagd. Het systeem is aangesloten op een High Speed modem, vergeleken met de transmissiesnelheid die een directe Internet link biedt misschien niet geweldig, maar veel beter kan het niet over de gewone huis- tuin- en keuken PTTlijn. De beheerder is Frans Goddijn. FGBBS is te bellen op 026 321 70 41.

4 NTG's winkel

Via de NTG is beschikbaar:

- **4allTEX CD-ROM:**
  Ruim 1 Giga byte aan ondermeer 4TEX utilities, fonts, TEX/LATEX/METAFONT/etc documentatie, de volledige MAPS 1 t/m 14, discussielijsten TEX-NL, TEXHAX, UKTEX van de afgelopen 5 jaren, etcetc. Kosten inclusief het uitgebreide 4TEX manual: ƒ 60,–.
- **Syllabus Advanced TEX course:**
  Insights and Hindsights, David Salomon (*revised*; ruim 500 pagina's). MAPS'92 speciale uitgave. Kosten ƒ 50,– voor leden en ƒ 60,– voor niet-leden (extra verzendkosten: ƒ 10,–).
- **PR set MAPS'93 speciale uitgave:**
  Ruim 25 pagina's; 1 exemplaar gratis voor leden; extra exemplaren: ƒ 2.50; niet-leden: ƒ 5,– (extra verzendkosten: ƒ 5,–).

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

5 MAPS 97.1

Sluitingsdatum voor het inleveren van artikelen, bijlagen, en/of mededelingen voor de volgende MAPS uitgaven is:

15 maart '97 (MAPS 97.1) en 15 september '97 (MAPS 97.2).

Aanleveren kopij voor de komende MAPS:

- *Bij voorkeur* in LATEX gebruikmakend van de: `maps.sty` of `maps.cls` Deze stijlfile is via de redactie te verkrijgen en beschikbaar op de TEX-NL fileserver, archive.cs.ruu.nl (ftp-site) en FGBBS (026 321 70 41).
  Daarnaast kunnen bijdragen ingestuurd worden gemaakt met `ltugboat` of `article/report` style/class files.
- Verder zijn bijdragen vanzelfsprekend ook welkom in *plain*-TEX of ongeformatteerd.
- Plaatjes bij voorkeur als (Encapsulated) PostScript file.

Bij onduidelijkheid gaarne kontakt opnemen met de redactie.

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

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

Bijdrage kunnen gestuurd worden naar:

`w.dol@lei.dlo.nl`

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

Van de Voorzitter

Erik Frambach

In 1995 dacht ik nog dat dat wel een extreem druk jaar was. Het bruiste van nieuwe ontwikkelingen en activiteiten. De NTG droeg daar ook flink aan bij, onder andere door het organiseren van de Europese TEX-conferentie. 1996 zou voor de NTG veel rustiger worden want je kunt nu eenmaal niet elk jaar pieken. Ook de gecombineerde EuroTEX/TUG-conferentie in Rusland gaf meer rust en zou tegelijk een teken zijn van nog betere internationale samenwerking. Op cd-rom-gebied zou het even rustig zijn want het 4allTEX-team kon voorlopig op zijn lauweren rusten.

Toch is het geen rustig jaar geworden, en eigenlijk ben ik daar blij om. In het voorjaar werden we onverwacht verblijd met een bezoek van Donald Knuth, waar we natuurlijk een extra bijeenkomst voor organiseerden. De TEXlive cd-rom kwam uit zodat nu ook op Unix-systemen TEX eenvoudig geïnstalleerd kan worden. Bovendien was dit voor velen de eerste kennismaking met een functionerende TDS (TEX Directory Structure). Wellicht is dit een belangrijke stap op weg naar verdere standaardisatie. In een artikel elders in de MAPS merkt Hans Hagen terecht op dat standaardisatie in de TEX-wereld nog ver te zoeken is. Daardoor zijn er nog massa's problemen niet opgelost. Je zou kunnen zeggen dat TEX nog steeds niet 'af' is want het blijft groeien, en gelukkig niet alleen in kwantiteit. Maar je zou ook kunnen zeggen dat TEX maar niet volwassen wil worden en dan is de vraag terecht of TEX nog wel toekomst heeft, verwend als we zijn met kant-en-klare pakketten, Plug'n'Play, gebruikersvriendelijkheid, WYSIWYG etc. Kortom, werk aan de winkel!

Dat er hard gewerkt wordt bleek ook weer eens op de conferentie in Dubna. Daar werden twee gloednieuwe produkten getoond die bewijzen hoe snel TEX kan inspelen op nieuwe ontwikkelingen. Ik doel op de programma's TEX2PDF en DVIPDF. Voor zover ik weet is er nog geen enkele tekstverwerker op de markt die rechtstreeks PDF kan genereren als output. Altijd gaat dat via extra (commerciële) programmatuur van Adobe, met name Acrobat Distiller. Het automatisch opnemen van links in PDF-bestanden (bv. via hyperref) is dacht ik nog steeds uniek. Met TEX zijn dus nu niet alleen typografisch aantrekkelijke documenten te maken maar ze kunnen ook interactief gemaakt worden. Ik geloof dat de relevantie van dat laatste, ondanks de Internet-hype, nog steeds zwaar onderschat wordt, terwijl TEX hier unieke functionaliteit biedt. Nog wel, tenminste. Met LATEX2HTML liep TEX ver vooruit, maar tegenwoordig kunnen bijna alle moderne tekstverwerkers HTML uitvoeren, en aan PDF-links wordt waarschijnlijk al gewerkt.

Ook op kleinere schaal wordt aan de weg getimmerd. Sinds kort is de Nederlandse spelling veranderd. Ofschoon de oude afbreekpatronen desondanks goed functioneren, kan het altijd beter. En de NTG zou de NTG niet zijn als daar geen aandacht aan besteed werd. Vandaar dat een groepje enthousiastelingen de nobele taak op zich heeft genomen een geheel nieuwe Nederlandse woordenlijst te bouwen met ruwweg de volgende functies: nieuwe afbreekpatronen voor TEX, en een woordenlijst voor spellingcontrole. Geheel in de traditie van de TEX-gemeenschap zal deze woordenlijst aan het publieke domein geschonken worden. Elders in de MAPS staat het project uitgebreid beschreven.

Binnen het NTG-bestuur is de dynamiek ook groot. Begin dit jaar hebben we afscheid genomen van Johannes Braams, en Hans Hagen en Taco Hoekwater welkom geheten als nieuwe bestuursleden. Wietse gaat nu het penningmeesterschap neerleggen. We hopen op heel korte termijn een nieuwe penningmeester aan te kunnen stellen, en vertrouwen er als vanouds op die uit de nog steeds groeiende NTG-geledereen te kunnen werven. De NTG is en blijft een vereniging van vrijwilligers en kan alleen gedijen als voldoende vrijwilligers zich daarvoor inzetten.

Ook dit najaar organiseert de NTG weer een bijzondere bijeenkomst, namelijk een een eendaagse cursus met als thema 'graphics'. Dankzij goede contacten met de Poolse TEX-gebruikersgroep GUST konden we hun Metafont/graphics-guru Bogusław Jackovski daarvoor strikken. Ook weer een initiatief dat aangeeft dat internationale contacten steeds sterker worden.

Dat bleek ook al op de Poolse TEX-gebruikersbijeenkomst in het voorjaar van dit jaar. GUST nodigt steeds meer buitenlandse gasten uit en doet als het ware EuroTEX nog eens dunnetjes over. Hebben we hier te maken met een tendens? Mogelijk gaan 'lokale' gebruikersgroepen steeds meer samenwerken en elkaar uitnodigen om elkaars bijeenkomsten bij te wonen. Zo zouden we een stuk of vijf mini-EuroTEX's per jaar kunnen hebben, en een heel grote conferentie voor de hele wereld. Dat lijkt me een aardige opzet. De financiële risico's van het organiseren van zo'n kleinere bijeenkomst zijn veel kleiner dan van een 'echte' EuroTEX-conferentie zodat ook kleinere groepen zoiets aankunnen. De bijeenkomsten kunnen korter zijn, bijvoorbeeld een of twee dagen. Dat is weer gemakkelijker in te plannen voor al die mensen met overvolle agenda's. Bovendien kan per bijeenkomst een thema gekozen worden, zodat geïnteresseerden specifiek naar die bijeenkomst kunnen gaan waar ze het meeste aan hebben. Op die manier zouden we mogelijk beter kunnen inspelen op de wensen van TEX-gebruikers.

De NTG bestaat nu bijna 10 jaar dus is het tijd om eens stil te staan bij uitgangspunten en doelstellingen van de club. Immers, er is enorm veel veranderd in die periode. De NTG blijft een vinger aan de pols houden en probeert op elke nieuwe ontwikkeling adequaat in te spelen, zodat we in 1998 op een grootse manier ons jubileum kunnen vieren, met vertrouwen in de toekomst.

Van de MAPS editor

Wietse Dol

Zucht! hij is weer af. Het maken van de MAPS is elke keer weer een bevalling. Met een draagtijd van nog geen maand, zonder de mogelijkheid tot een keizersnede betekent dat er in nog geen twee weken tijd de MAPS met veel gepuf en gesteun gemaakt is. Natuurlijk is 4TEX weer de steun en toeverlaat die het mogelijk maakte om de MAPS op tijd uit te laten komen. Dit keer is de MAPS gemaakt op een pentium notebook met LATEX2ε en de Windows'95 frontend van 4TEX.

Zoals gebruikelijk allereerst de dankbetuigingen aan alle mensen die aan deze MAPS hebben bijgedragen: de auteurs van de diverse artikelen en de redactie. Aan alle auteurs maar vooral ook alle andere mensen blijf ik zeggen: schrijft artikelen en stuur ze naar ons. Als redactie kunnen we nu met winter reces en dankbaar genieten van een paar maanden rust.

Afgelopen jaar is wederom een goed MAPS jaar geweest. De twee publikaties zijn weer goed gevuld en voor een ieder (beginner en/of TEXneut) is er elke keer weer voldoende te lezen geweest. Ook dit nummer zijn er weer artikelen waar u uw vingers bij af kunt likken.

In dit nummer zult u zien dat er van diverse kanten een poging gedaan wordt om een discussie te starten over de toekomst van TEX. Naar mijn mening wordt er vaak bedoeld: de toekomst van gebruikersgroepen. Als redactielid zal ik van deze kolom gebruik maken om eens mijn standpunt hierover toe te lichten. De afgelopen jaren zie je het ledental van alle gebruikersgroepen dalen (ja ik weet het! de NTG bleef de afgelopen jaren licht groeien, maar deze groei is er uit en ik voorspel dan ook een donkere toekomst, tenzij...). Dit is voornamelijk het gevolg van het succes van TEX! Was men vroeger bijna genoodzaakt om lid te worden van een gebruikersgroep (om informatie te krijgen) is het vandaag de dag een makkie om alle noodzakelijke informatie te krijgen. Gratis discussie lijsten, goedkope en leerzame boeken en tegenwoordig diverse TEX CD-roms maakt het lid zijn/worden van een gebruikers groep steeds minder noodzakelijk. Natuurlijk zijn alle in de vorige zin genoemde dingen het resultaat van de activiteiten van de diverse gebruikersgroepen, maar daar zal een gemiddelde TEX gebruiker niet wakker van liggen. Als het moet betaalt ie gewoon wat meer geld voor zijn gewenst produkt. De discussie die dan ook gevoerd moet worden is volgens mij niet over de toekomst van TEX maar *wat kunnen de gebruikersgroepen de diverse TEX gebruikers bieden*. Naar mijn mening dient er dan ook door de gebruikersgroepen gekeken te worden naar wat voor soort gebruikers er in de wereld zijn (wist u dat er in Duitsland zo'n 150.000 gebruikers zijn waarvan er slechts 2.000 lid zijn van Dante, gezien de CD-rom verkopen denk ik dat in Nederland dezelfde verhoudingsgetallen gelden en je snel kunt uitrekenen dat er zo'n 22.500 gebruikers in Nederland vertoeven). Daarna moet gekeken worden wat de wensen van deze gebruikers zijn en hoe een gebruikersgroep daaraan tegemoet kan komen. Omdat de computerwereld vandaag de dag zeer klein is lijkt een samenwerking tussen alle gebruikersgroepen essentieel. Vergelijk het maar met de firma Microsoft die in elk land zijn eigen afdeling heeft maar als wereldfirma zijn visie en toekomst bepaald. De lokale afdeling verzorgt/vertaald deze wereldvisie naar de landelijke gebruikers (en de land specifieke wensen). Conclusie van mij is dan ook dat de gebruikersgroepen alleen kunnen overleven als ze gaan samenwerken. Deze samenwerking moet in eerste instantie betrekking hebben op het uitgeven van 1 gezamenlijke periodiek. Dus vaarwel MAPS en hallo de *WorldBoat*. Natuurlijk (net als bij Microsoft) moet het wel mogelijk blijven om de diverse artikelen in de WorldBoat te vertalen naar de lokale taal. Door deze samenwerking kan er een veel beter en goedkoper produkt worden geleverd met veel minder inspanning.

De afgelopen jaren is gebleken dat de MAPS de belangrijkste reden is voor de gebruikers om lid te zijn van de NTG. Als je dan ziet dat 90% van deze MAPS door slechts twee redacteuren is gemaakt, dan kun je wel problemen voorspellen als deze twee mensen er mee ophouden. Nu dat is min of meer zover. Gerard heeft vorig jaar zijn functie als hoofdredacteur neergelegd en na deze MAPS zal ik niet langer meer de hoofdredacteur zijn. De belangrijkste reden is wel het buiten proportioneel beslag wat de MAPS op iemand legt. De afgelopen twee jaar ben ik in mijn vrije tijd niet verder gekomen dan TEX zaken. De rek is er dan op een gegeven moment eruit. Natuurlijk zullen Gerard en ik actief blijven als redactielid maar zullen we de kar niet meer trekken. Wie dat wel gaat doen is op het moment van schrijven onbekend. Dit brengt mij tot een tweede kritiekpunt richting NTG: hoeveel NTG leden zijn actief? Als ik kijk naar de ontwikkelingen en bijdragen aan TEX dan is Nederland (en dus ook de NTG) vaak afwezig. Zelf voor het schrijven van vele leerzame een boeiende overzichts artikelen moest de MAPS redactie altijd terugvallen op mensen in het buitenland. Bekijkt u eens hoeveel verschillende mensen de afgelopen vijf jaar de MAPS gevuld hebben en hoeveel daarvan Nederlander zijn. Ja inderdaad haal u zakdoek maar tevoorschijn, maar bovenal *DOE ER EENS WAT AAN, WORDT ACTIEF*. De toekomst van de gebruikersverenigingen (en dus indirect de toekomst van TEX) ligt in *UW* hand.

Concept begroting van de Nederlandstalige TEX Gebruikersgroep

voor het jaar 1997

Inkomsten		**Uitgaven**	
Contributie	ƒ 23.000,00	Administratie	ƒ 1000,00
Rente	ƒ 600,00	Kamer van Koophandel	ƒ 150,00
		Bijeenkomsten	
		Bestuurskosten	ƒ 750,00
		Computerfaciliteiten	ƒ 125,00
		Nieuwsbrief/Verslagen	ƒ 16.500,00
		Reis bijdragen	ƒ 1.500,00
		Representatie	ƒ 200,00
		Afschrijving	ƒ 1.182,00
		Sponsoring	ƒ 1.000,00
		Bijdrage Bursary fund	ƒ 1.000,00
		Onvoorzien	ƒ 193,00
	ƒ 23.600,00		ƒ 23.600,00

Hierboven vindt U de voorlopige begroting voor 1997 van de Nederlandstalige TEX gebruikersgroep. Een toelichting volgt hieronder.

Toelichting

- **Inkomsten**:
  1. Contributie
     De post contributie is gebaseerd op het aantal betalende leden in augustus 1996 en op een contributie van ƒ 90 (NTG lid)/ƒ 245 (instituuts lid). De ledenstand van "augustus" 1996 was:

31	instituten	31	× ƒ 245,00	ƒ 7.595,00
4	aanvulling	4	× ƒ 65,00	ƒ 260,00
9	studenten	9	× ƒ 65,00	ƒ 585,00
111	personen	111	× ƒ 90,00	ƒ 9.990,00
66	personen	66	× ƒ 81,00	ƒ 5.346,00
				ƒ 23.776,00

  2. Rente De vereniging heeft in de afgelopen jaren een klein kapitaal opgebouwd. Als dit niet nodig is om een tegenvaller op te vangen is het mogelijk behoorlijk wat rente te krijgen. De rente is behoorlijk lager begroot dan in 1996 omdat deze rente inkomsten mede gebaseerd was op de rente van CD-rom inkomsten. Omdat de CD-rom verkopen een aparte rekening heeft gekregen (buiten de boekhouding van de NTG) zijn de rente inkomsten van de NTG rekening fors lager begroot dan in 1996.

- **Uitgaven**:
  1. Administratie Dit is bedoeld voor materiaal voor de secretaris en penningmeester. De hoogte is bepaald aan de hand van de hoogte van de uitkomst over 1995 en de realisatie in 1996 tot nu toe.
  2. Kamer van Koophandel Dit is een jaarlijks terugkerende inschrijving van ƒ 61,00 en elke verandering van bestuur brengt extra kosten met zich mee.
  3. Bestuurskosten Hieronder vallen kosten als telefonische vergaderingen, vergoeding reiskosten voor een eventuele physieke bijeenkomst etc.
  4. Computerfaciliteiten Het kunnen gebruiken van de `NTG.NL` domain name kost jaarlijks geld.
  5. Nieuwsbrief/Verslagen Het copiëren en verspreiden van de verslagen van de bijeenkomsten. De kosten bedragen ongeveer ƒ 22,50 per exemplaar. Er is rekening gehouden met twee maal een oplage van 350 exemplaren en een twee maal verschijnen per jaar.
  6. Reisbijdragen Het is de bedoeling dat de vereniging bijdraagt in de kosten van het bijwonen van buitenlandse bijeenkomsten die met TEX te maken hebben. In 1997 moet rekening gehouden met het bijwonen van de EuroTEX bijeenkomst in Spanje (Barcelona).
     Als tegenprestatie wordt een verslag van de bijeenkomst verwacht, ter publicatie binnen de vereniging.
  7. Bijdrage Bursary fund In 1993 is besloten jaarlijks een bijdrage te leveren aan het Bursary fund van TEX conferenties. Dit fonds is bedoeld om TEX gebruikers met financiële problemen in de gelegenheid te stellen een internationale TEX bijeenkomst bij te wonen.
  8. Afschrijving In 1994 is voor het secretariaat een laserprinter aangeschaft. Deze wordt over drie jaar afgeschreven.
  9. Sponsoring Het bestuur is van plan in 1997 een financiële bijdrage te leveren aan het bulletin board van Frans Goddijn (ƒ 1000,00).
  10. Representatie Als bestuursleden van zusterverenigingen bij onze bijeenkomst uitgenodigd worden, kan een tegemoetkoming in de kosten worden gegeven.

Een rol voor TEX in het 3de millennium

Michel Goossens, TUG president

De zomer 1996 hoort al tot de wereld van onze goeie herinnneringen, en we zijn allemaal weer vlijtig aan 't werk. Van de jonge studentin, de wakkere onderzoeker tot de gepensioneerde doorzetter. Maar niettegenstaande onze verschillen in educatieve specialisering, professionele bezigheid of culturele achtergrond, er is dat unieke wat ons, lezers van MAPS immer en altijd weer stimuleert, namelijk onze bewondering voor Knuth's meesterwerk TEX. Niettegenstaande TEX bijna twintig jaar geleden werd ontwikkeld, is het nog steeds een kwaliteitsreferentie waar de meeste hedendaagse tekstverwerkers een voorbeeld kunnen aan nemen wat betreft de mogelijkheden om zijn gedachten keurig en precies, visueel aantrekkelijk en duidelijk op papier te zetten.

TEX is perfect meertalig en wordt thans gebruikt voor het schrijven in vrijwel alle talen op onze blauwe planeet. Dit werd bijzonder klaar aangetoond in juli van dit jaar op TUG'96 in Doebna (Rusland) waar meerdere voordrachten dit belangrijke aspect benadrukten: TEX is nu dagelijkse koek in de diverste uithoeken van Rusland voor het produceren van boeken in een zestigtal verschillende talen geschreven in het Cyrillisch alfabet. Laat me van de gelegenheid gebruik maken om $\mathcal{NTG}$ hartelijk te danken voor de financiële steun aan deze TEX Users Group conferentie. De aanwezigheid van NTG's stichter-voorzitter Kees van der Laan en van de huidige voorzitter Erik Frambach werd ook erg op prijs gesteld.

Maar terugkerend naar mijn betoog zou ik willen beklemtonen hoe we dank zij LATEX de structurele informatie aanwezig in onze documenten kunnen uitbuiten om teksten een nieuw leven in te blazen voor hun gebruik in de hyperwereld van de Internet. Aan de hand van vertaalprogramma's die LATEX omzetten in SGML/HTML en PDF (of omgekeerd) kunnen we een optimale bladzetting combineren met alternatieve voorstellingsmogelijkheden om onze informatiebronnen globaal en wereldwijd beschikbaar te stellen.

Als er iets opvalt de laatste jaren dan is het wel hoe snel ontwikkelingen op het gebied van de informatica, en in het bijzonder het elektronisch document, zich opvolgen, en hoe verschillende takken op elkaar inspelen, elkaar mutueel beïnvloeden en in een ware symbiose leven. Het geschreven woord dreigt verloren te gaan in een wervel van visuele (kleur, reliëf, animatie, virtuele realiteit) en auditieve (stereo, zelfs quadrafonie) effecten die *ad nauseam* de beeldbuis verblinden. Er gaan al stemmen op om al deze effecten toegankelijk te maken via „`\speciale`" uitbreidingen van de TEX syntax en om normen te definiëren om Netscape netsurfen en PDF bookmarks te rationaliseren.

Is dit het einde van ons geliefd en sober TEX universum of is het een voorbijgaande manie; moeten we ons aanpassen of standhouden zoals de kapitein op de brug van de zinkende Titanic?

Ik denk dat we best alle materiële media assimileren en de verschillende communicatiemiddelen combineren om zo een optimale synthesetaal op te bouwen. Aan de hand van *plug-and-play* CDROMs moeten we trachten het TEX paradigma te laten binnendringen in de met windows-vergroeide huiskamers om zo een TEX-cultuur op te bouwen. Ontwikkelingen van programma's zoals `dvi2pdf` of `tex2pdf`, die ons toelatex TEX-bestanden in PDF om te zetten, moeten gesteund worden. Een initiatief zoals Omega, een 16-bit uitbreiding van TEX die zich baserend op the internationale Unicode norm moet aangemoedigd worden gezien het toelaat complexe teksten te zetten in alle mogelijke talen, zij het die met alfabetische, syllabische of pictorale schriften.

Al deze ontwikkelingen vinden echter niet in een vacuüm plaats, maar worden permanent gestimuleerd door de positieve en creatieve samenwerkingsgeest die de TEX wereld karakteriseert en waar altruïsme, beschikbaarheid en collegialiteit de trefwoorden zijn.

„Verdomd idealisme !", hoor ik jullie al roepen. Wel nee, dat is wat TEX Users Group, $\mathcal{NTG}$, samen met andere TEX gebruikersgroepen al jarenlang proberen te realiseren. Dat is waarom we er in TEX Users Group nog steeds mee door gaan, niettegenstaande een niet al te rooskleurige financiële situatie, te wijten aan een steeds afnemend aantal leden en de hogere lasten (papierprijs, verzendingskosten van TUGboat, enz.). We zijn volop bezig een nieuw model voor TEX Users Group als functionele organisatie op te bouwen. De vernieuwde TEX Users Group moet zich beter inwerken in de Internetwereld waarvan we onherroepelijk deel uitmaken. Deze meer expliciete aanwezigheid op het „Net" moet ertoe bijdragen TEX Users Group bekend te maken in bredere kringen en vormt zo een ideale wervingsbasis voor nieuwe TEXgebruikers. TEX Users Group is tevens van plan een centrale en gemakkelijk toegankelijke WWW gegevensbank voor alle met TEX in verband staande informatie op te bouwen en up-to-date te houden.

Dat zijn onze plannen voor TEX Users Group. En ik hoop dat we bij de realisatie ervan terecht kunnen bij onze collega's van $\mathcal{NTG}$, GUTenberg, enz. We stevenen recht af op het derde millenium en we zullen er aankomen vóór we het weten. Of TEX er nog de rol zal spelen waar het recht op heeft hangt voor het grootste gedeelte enkel van ons, TEXgebruikers, af. Ik kan enkel wensen dat TEX Users Group niet alleen zal staan om in het jaar 2000 TEX met glorie de drempel van de 21ste eeuw te laten overschrijden.

(Cyr)TUG 96, and some more

Kees van der Laan & Erik Frambach

Perfect, just one word. 'Atlichna,' dear Russian friends. Have you ever attended a TUG meeting and welcomed by real fire-works? Well, the Russians did.

A social event? As said the very first evening, while strolling along the Volga promenade, we had a disco and fire-works, to start with, to come in the right mood. The next evening we were welcomed officialy via a welcome dinner à la Russe, with many a toast and a dance or two afterwards, 'kanechna,' dear Russian friends.

The trip to Sergijev Posad was great and the flat tyre on the return was there to test our survival capabilities. Ever heard of a survival outing at a TUG meeting? Well, the Russians did.

And what about the trip to the old town Kimra while it was raining cats and dogs? The local museum devoted to life through the ages could be visited at leisure. Then there was the boattrip under a blue sky up and down the Wolga, with the picnic and delicious 'shashlik' afterwards with ample drinks, not to forget the Vodka and wine.

Buying souvenirs in Russia can be strainuous. Not during this conference however. Each morning next to the conference room door souvenirs were offered, to ease the looking for them by the participants. Last but not least there was the guided tour through Moscow. Thank you friends, no travel agency could have done better. How Should I ever have known about the 'Poklony Gora?' As said 'atlichna.'

But, ... it's not about a tourist trip, a TUG meeting was at stake. So let us go back to business.

A very modest TUG meeting with 80 odd participants. During the opening ceremony Irina Makhovaya depicted lucidly the coming of age of CyrTUG. Within three years the group has matured. Bravo! CyrTUG has solved the problems of adapting TEX to handle cyrillics, and enjoys outstanding TEXies as members. Sasha Berdnikov earned the TUG price for his contributions. Do we have to say more? The group has 700 members with Knuth as honorary member 314.

A snapshot of the contributions

Yannis Haralambous' talk about Arabic an TEX was a bit too much. He explained a lot about Arabic characters and typography, nothing about TEX or METAFONT.

Next Sergej Znamenskii described the various font encodings that are currently being used for cyrillic.

Then Olga Lapko explained that cyrillic $\neq$ cyrillic, but many languages. She also showed that even in the Unicode table some characters are missing.[1]

Karel Píška showed how cyrillic alphabets are ordered, and explained what problems are involded with that.

The DC fonts are moving towards stability and completeness. Jörg Knappen presented version 1.3 of these fonts, in which several details have been improved, such as accents on capitals.

Jörg also presented a paper by Fukui Rei about TIPA, a system for processing phonetic symbols in LATEX. The package includes a complete rewrite of METAFONT sources and macros, based on the old 7 bits version.

In the evening there was the dinner party with lots of good food, vodka, wine, toasts and speeches. Gabriel Valiente Feruglio announced that he is willing to host the next meeting in Barcelona!

Yannis Haralambous then showed us a new development in the Ω system: the fonts ΩTimes and ΩHelvetica.

Unicode was also the theme of Richard Kinch's lecture. He explained the advantages of extending TEX to Unicode. He implemented Unicode features in his TrueTEX system.

Sasha Berdnikov demonstrated a new approach to CM fonts: they can be implemented as Multiple Master fonts, so any degree of (sans-)serifeness or boldness can be obtained by abstracting from the 70 odd METAFONT parameters that define the shape of the characters.

Since Sasha has also done a lot of work in the field of virtual font management his next lecture was on an update of the program VFComb. The documentation is also available in English now on CTAN.

Interplatform compatibility is a serious problem, even with TEX documents, because of different code pages that people use. Peter Ovchenkov explained the problems specific for cyrillic texts.

Then Dag Langyhr revealed his project 'StarTEX' which is an HTML-like shell around TEX-commands. It's an educational markup language that protects students against unintended use of unknown commands. It will also give more sensible error messages, and even do some error recovery.

Gabriel Valiente Feruglio made a survey of scientific journals that accept LATEX submissions, and explained the pros and cons of electronic submission of papers.

TEX and related programs are not for the faint hearted. Sergej Znaminskii showed us a new approach with a user friendly interface, using some built-in 'intelligence'.

The Russian typographer Mikhail Grinchuk introduced us Russian traditions of typesetting. Many details such as punctuation, quotes, cdots, ldots and dashes are different from 'the west'. Some of them are very hard (impossible?) to implement in TEX.

[1] It is rumoured that the CyrTUG people did solve their problems with cyrillics during this conference. The following anecdote is attributed to Oleg Khristenko: 'Olga Lapko should come up with a new font encoding or had to marry him.'

'DVI-based electronic publication' is the way to go, according to Laurent Siebenman. He evaluated several potential electronic formats and found the dvi format to be most suited, although some problems still have to be solved (e.g. fonts, graphics).

Sergej Strelkov explained how to make indexes for VINITI's Mathematical Abstract Journal. He uses LaTeX2.09, Perl, MakeIndex and DVIspell in a rather complex scheme that was quite difficult to follow. Fortunately, most of the process is automated.

'Graphics and TeX' is a theme on almost any TeX meeting. Now it was Kees van der Laan's turn to show that, as he puts it, 'a child can do it', using his Turtle Graphics macros.

Kees is also involved in producing graphics by METAFONT and MetaPost, and he showed us some beautiful examples generated from remarkably elegant code.

Petr Sojka then informed us about the latest developments in the project of adapting TeX to write PDF instead of DVI output. His DVI2PDF program is based on change files to the original TeX WEB sources. The program is in α-test now (July 1996). Some features are not yet implemented (e.g. the 'hyperref' links, bitmapped fonts and graphics), but will certainly be added later. More information is available from `http://www.cstug.cz/~thanh/tex2pdf`.

PDF is a hot issue. Sergej Lesenko showed us another approach to generating PDF. His program DVI2PDF is based on Thomas Rockiki's DVIPS. It already features rotated and scaled text, graphics (BMP and JPEG), annotations and bookmarks, partial font loading, and color. EPS graphics are not yet supported, and neither are bitmapped fonts and thumbnails. No report in the (pre)proceedings but it will appear later.

Yurij Ivanov then gave a lecture on text processing at JINR, the Joint Institute for Nuclear Research, which was hosting the conference. Needless to say that they use TeX a lot and are very happy with it.

Andrej Slepukhin showed us methods of multilingual text processing.[2]

At several other TeX meeting Michel Goossens has already shared with us his knowledge about the process of converting LaTeX into HTML and back. In his lecture about LaTeX and the Internet he informed us about several programs that are useful in this context. e.g. HTML2LATEX (originally from Nikos Drakos) and TYPEHTML (from David Carlisle, which enables TeX to typeset HTML directly).

Andrej Astrelin showed us his work on programming graphic objects (such as lines, arcs and splines) in C++. The relevance for TeX users still escapes us, as we now have so many alternatives like MetaPost and other drawing tools.

LaTeX's picture environment is powerful, but not perfect. Sasha Berdnikov extended the picture environment with useful macros and also pointed out some problems that still need to be dealt with, e.g. big arrows, double lines and dotted/dashed lines.

Kees van der Laan explained to the audience his BLUe system as a 'mental tool'. He gave an overview of its functionality, and announced that he has finished the user's guide – Publishing with TeX – next to the technical documentation. The Russian translation of the user's guide will be available in August.

Evgeni Pankriatev, the president of CyrTUG, spoke about incompatibilities between various Russian TeXs. CyrTUG is bound to solve these problems. They will concentrate on Russian versions of plain TeX, LaTeX, and AMSTeX, adapt the usual style files, next to documentation in Russian.
Interesting too is their project on graphic object representation and an implementation of a graphic library.
With respect to support CyrTUG plans a CTAN mirror and a network of consultants. As said before CyrTUG has matured.

Awards

At the closing session 4 people got awarded:

- Sasha Berdnikov c.s for his work on VFcomb, his mutiple master fonts and his extensions to the LaTeX picture environment;
- Dag Langmyhr for his refreshing approach implemented in StarTeX;
- Michel Goossens for his general work and especially for organizing this conference;
- Petr Sojka for his work on TeX2PDF.

During the conference dinner many a toast was given. Volkert Schaa on behalf of DANTE offered their CD-ROM snapshot of the CTAN to every participant. Gabriel Valiente offered to organize the next TUG meeting in Barcelona.

Sebastian Rahtz kept up with tradition by offering the Cathy Booth award. At the conference the traditional mug was there but T-shirts were missing. As real cooperating polyglots we volunteered to handle this after the conference.

Those who were not present this time have a second chance to experience a TeX meeting à la Russe by attending a CyrTUG meeting. The next one will probably be in St. Petersburg.

All-in-all a nice and very cosy – 'ouyoutna' the Russians would say – conference. Thank you all, especially the perfect host the Dubna Institute of Nuclear Research, with all its volunteers which made this meeting such a smashing success.

[2] The paper is missing and we are sorry that we can't give more details.

De spelling van het lot.
Over het ontstaan van een nieuwe NTG werkgroep

Frans Goddijn & anderen

fg@fgbbs.iaf.nl

23 Sept 1996

Abstract

Het SPELLINGBESLUIT gepubliceerd in Staatsblad 394 *(Besluit van 19 juni 1996, houdende voorschriften omtrent de schrijfwijze van de Nederlandse taal)* heeft niet alleen gezorgd voor veel gekrakeel tussen neerlandici, maar het vormde ook een buitenkans voor uitgeverijen van werken als HET GROENE BOEKJE (binnen zes maanden 500.000 verkochte exemplaren). Kleinere ondernemingen brachten floppies op de markt met woordenlijsten en algoritmen. Intussen sloeg een groepje mensen (bestaande uit — in alfabetische volgorde — Erick Branderhorst, Erik Frambach, Frans Goddijn, Hans Hagen, Gerard van Nes, Piet Tutelaers, Jacoline van Weelden en Peter van Zeeland, en terzijde gestaan door Hans Linders) de handen ineen om te komen tot een vrij beschikbare, zo correct mogelijke Nederlandse woordenlijst.[1]

Nederlands uit de USA

Nu de nieuwe spelling er is, wil ik graag een nieuwe, correcte Nederlandse woordenlijst om aan mijn spellingchecker te voeren. Ik heb geen nieuwe spellingchecker nodig en al helemaal geen tekstverwerker, ook al adverteren de verkopers ervan ermee dat hun product nu is uitgerust met HET GROENE BOEKJE.

Diverse leveranciers staan klaar om ons onze eigen taal te verkopen: UBIQ BV in Montfoort (0348-475385) heeft een floppy voor krap dertig gulden, echter wie dan denkt een woordenlijst te kopen, krijgt een verrassing: er zitten wel 160.000 woorden op de floppy maar je kunt er niet bij. Sterker, ze kunnen er zelf ook niet bij want, zo verklaart een medewerker van UBIQ BV, “de woordenlijst is gecomprimeerd en komt van NOVELL in Amerika. Hij is bedoeld voor gebruikers van WORDPERFECT die er hun spellingchecker mee kunnen updaten”.

Zijn we dan `*!#@^:=(` zover gezakt dat we de woorden van *onze eigen taal* moeten aanschaffen in de Verenigde Staten van Amerika?!

Hans Hagen schreef me: “Alle pogingen van commissies ten spijt, wordt taal gemaakt door de gebruikers. Het is dan ook te gek voor woorden dat diezelfde gebruikers moeten betalen om de door de overheid voorgeschreven woorden te mogen gebruiken.”

INP (Postbus 765, 2270 AP Voorburg) levert ook een oplossing, namelijk een floppy waarmee WP-gebruikers hun bestaande woordenlijst kunnen bijwerken. Niet alle woorden zijn veranderd: er moeten volgens INP ongeveer 1200 woorden weg en er komen zo’n 4800 woorden woorden bij om de WP-gebruiker in de hedendaagse spelling te laten schrijven.

Verder kan ook bij de STICHTING SOCIALE DATABANK NEDERLAND een verbeterde Nederlandse WP-woordenlijst besteld worden. Zij melden dat t.o.v. WORDPERFECT 5.1 ruim 7.000 fouten zijn verbeterd, en bijna 112.000 ontbrekende woorden zijn toegevoegd. Leuk voor de WP-gebruiker, maar ik stel nu eenmaal wat hogere eisen aan mijn zetwerk dan met WP mogelijk is.

Fouten bij WP

Erik Frambach heeft destijds bij het vervaardigen van de woordenlijst in de oude spelling gebruik gemaakt van WORDPERFECT 5.1 ter controle. Maar wat bleek… het was niet zozeer WORDPERFECT waarmee de fouten uit zijn eigen woordenlijst-in-wording werden gehaald:

> “Ik ben me daar hardstikke doodgeschrokken van de massa’s fouten (in WORDPERFECT). Wat erg dat zoveel mensen dit als referentie gebruiken! Naar aanleiding daarvan heb ik het volgende stukje geschreven in ons facultair computerblaadje:
>
> **De niet zo perfekte WordPerfect woordenlijst**
>
> Tot voor kort gebruikten velen voor het controleren van de spelling van de Nederlandstalige TEX-documenten de woordenlijst die bij WORDPERFECT 5.1 wordt geleverd. Deze lijst is echter erg beperkt van omvang en inconsistent wat betreft welke verbuigingen en meervouds- en enkelvoudsvormen zijn opgenomen. Het ergste is echter dat de lijst *honderden* fouten bevat (nee, ik overdrijf niet).

[1] Dit is een bewerking van een artikel uit het septembernummer van Computer Info. De kopij van het artikel is eerst door Erik Frambach voorzien van diverse links naar email-adressen, homepages en ftp-sites en de tekst is later door mij uitgebreid.

- Er staan typfouten in: bv. ‘o’ wordt ‘i’, ‘m’ wordt ’n’. (Een veel voorkomende fout bij tikwerk, aangezien de ‘o’ en de ‘i’ naast elkaar zitten op het toetsenbord evenals de ‘m’ en de ‘n’).
- Er zijn letters vergeten.
- Er zijn letters omgedraaid.
- Woorden worden ernstig verminkt: ‘belastingtruc’ wordt ‘balstingtruc’.

Geschrokken van de onbetrouwbaarheid van deze lijst ben ik zelf op zoek gegaan naar betere alternatieven. Uit ‘public domain’ woordenlijsten heb ik een nieuwe, ruim twee keer zo grote lijst samen kunnen stellen die vele malen beter is.
WORDPERFECT Nederland heb ik aangeschreven om te vragen of ze interesse hebben, maar nee, zo schref Lisa Boerlage, de Teamleider Support van WORDPERFECT: *“Het is bij WordPerfect Nederland en WordPerfect Corporation bekend dat de Nederlandse woordenlijst beperkingen oplevert. Ook zijn wij ons bewust van het feit dat er in de lijst spelfouten voorkomen. Deze oneffenheden zijn door ons verzameld. Wij hopen dan ook in de volgende versie van* WORDPERFECT *een betere woordenlijst te kunnen leveren.*
Als je een fout vindt kun je die melden. Kennelijk is dit hoe WORDPERFECT denkt de woordenlijst te verbeteren. Maar zo komen we er nooit. De lijst is slecht opgezet en zal niet wezenlijk verbeteren door hier een daar een woord aan te passen.
Vertrouw de Nederlandse spelling van WP niet! Ook niet in volgende ‘verbeterde’ versies. Wie werpt eens een kritische blik op de andere talen?”

Dat schreef Erik Frambach in januari 1993, en toen WP 6.0 uitkwam ging hij eens voorzichtig kijken of er iets verbeterd was. Ja er was iets verbeterd, maar hij kon ook weer moeiteloos massa’s oude fouten aanwijzen.

Overigens is het vaker onthutsend om kritisch te kijken naar woordenlijsten die algemeen als ‘de’ standaard worden gezien, zoals verderop zal blijken…

ASCII gewenst

Het is trouwens de vraag of het voldoende is om simpelweg een aantal duizenden woorden uit een oudere woordenlijst te vervangen, zo redeneerde Hans Hagen:

“Onze taal kenmerkt zich door een grote hoeveelheid woorden. Dit is een direct gevolg van het feit dat veel woorden te combineren zijn tot nieuwe. Deze combinaties leiden vervolgens weer tot afspraken over de verbindende letters, zoals een ‘n’ of een ‘s’. Als de volledige woordenlijst van de Nederlandse taal vele meters beslaat en HET GROENE BOEKJE maar enkele centimeters dik is, hoe weet je dan zeker of iets goed geschreven is? De technisch wetenschappelijke wereld heeft daarnaast nog een geheel eigen woordenschat, aangevuld met wederom een veelheid aan combinaties. Gelukkig is deze wereld uitstekend in staat de eigen hulpmiddelen te creëren om teksten op spelling te controleren, maar waar haal je de juiste spelling vandaan?
In de TEX-wereld bijvoorbeeld, waar een gebruiksvriendelijke interface bij de invoer van tekst veel minder belangrijk wordt gevonden dan de typografische kwaliteit van de uitvoer, wordt hoofdzakelijk gebruik gemaakt van ASCII teksten. Bij het controleren op de juiste spelling worden eveneens in ASCII vastgelegde woordenlijsten gebruikt. De gecodeerde woordenlijsten van commercile tekstverwerkers, zoals WP en MS-WORD zijn daarbij onbruikbaar. Commissies die een wijziging in de spelling voorstellen zonder zorg te dragen voor voor ieder toegankelijke bestanden in ASCII formaat, schieten dus deels hun doel voorbij.”

Falende algoritmen

Maar zelfs voor WP-gebruikers is het niet alles goud wat er blinkt. “De spellingchecker van Prisma is voorlopig niet aan te raden”, zo schreef Henk Langerak van het Algemeen Dagblad. In elk geval niet in WP 6.0a en 6.0b. Daarin gooit-ie heel veel woorden weg. Hij herkent veel woorden niet en zeker niet de nieuwe spelling.”

Joost Bloemsma van uitgeverij Het Spectrum reageerde hierop:

“Deze spellingchecker is niet in de handel geweest omdat de tekortkomingen door ons bemerkt werden. Hij was overigens identiek aan die van UBIQ. Deze spellingchecker bevat niet een gecomprimeerde woordenlijst maar een verzameling algoritmen die de spelling moeten controleren, en daar zit nou de kneep: deze nieuwe spelling is zo inconsistent dat ze niet in een verzameling algorithmen te vangen is. De oplossing zou een schier eindeloze lijst van woorden zijn, zie De Nieuwe Spellinggids (Prisma/Van Dale/Wolters) met meer dan 275.000 woorden, zonder verbuigingen en vervoegingen (slechts fl 29,90). Een dergelijke lijst als spellingchecker is echter onaanvaardbaar langzaam.”

Ik heb gevraagd of deze Nieuwe Spellinggids ook op floppy is te verkrijgen, maar dat bleek niet het geval te zijn. Van Dale Lexicografie verwees me naar uitgeverij Het Spectrum waar Joost Bloemsma werkt. Hem heb ik de tip gegeven dat zo’n woordenlijst in samenwerking met een spellingchecker als AMSpell zeker niet *onaanvaardbaar langzaam* is, doordat de woordenlijst voor gebruik wordt gecompileerd en gendexeerd. Het Spectrum zou, met deze spellingchecker, woordenlijst en vooral hun marketingapparaat, eenvoudig een goedkope spellingfloppy op de markt kunnen brengen.

Het Groene Boekje

SDU (070-3789911) is marktleider met HET GROENE BOEKJE (“onvolledig, onsystematisch en gebruiksonvriendelijk” aldus Onze Taal), maar de beoogde lancering in januari van de elektronische versie van het Groene Boekje werd uitgesteld tot juni dit jaar. Oorzaak hiervan was het grote aantal verschillen tussen dit boekje en de Dikke Van Dale. Die moesten eerst zoveel mogelijk worden weggewerkt. De kopers van het ELEKTRONISCH GROENE BOEKJE zijn daarmee in het voordeel vergeleken bij de mensen die het papieren boekje kochten. Overzichtelijk is het zeker en systematisch is het ook. Gebruiksvriendelijk zou je het kunnen noemen als je ziet hoe de SDU erin is geslaagd de woorden van onze taal te verpakken in een

user interface als een etalageruit: je kunt de woorden op allerlei manieren bekijken maar je kunt er niet 'bij'.

Ben je aan het schrijven en twijfel je aan de spelling van een woord, dan kun je het ELEKTRONISCH GROENE BOEKJE starten en je zoekt informatie over het woord. Het ELEKTRONISCH GROENE BOEKJE is echter traag, erg traag: een woord vinden kost enkele tot vele seconden, bijna even lang als het kost om hetzelfde woord in de papieren versie op te zoeken. Wachten of bladeren, het is kiezen tussen twee kwaden. Een heel document op deze manier controleren is onbegonnen werk.

Je kunt met het ELEKTRONISCH GROENE BOEKJE ook dictees maken en met anagrammen goochelen. Ook kun je een verhaal over de nieuwe spellingsregels lezen, maar n ding kun je niet en dat is wat mij betreft het enige van serieus belang: een leesbare lijst van woorden genereren.

En dat terwijl er *gratis* spellingcheckers zijn die beter zijn dan die van WP!

Alternatieve spellingcheckers

AMSpell bijvoorbeeld, geschreven en onderhouden door Arjan Merckens en Erik Frambach, naadloos ingepast in de 4TEX-structuur, met woordenlijsten voor Nederlands (oude spelling), Engels, Amerikaans, Frans, Duits, Italiaans en Spaans.

Bij PRAGMA (038-4229775) heeft men een spellingchecker TEXSPELL gemaakt die gratis mag worden verspreid. Zelf gebruiken ze deze samen met de editor TEXEDIT, maar hij is ook afzonderlijk te gebruiken. Ik citeer Hans Hagen van PRAGMA:

> "Met TEXSPELL kunnen teksten worden gecontroleerd op foutief gespelde woorden. Men kan daarbij werken met verschillende lijsten, bijvoorbeeld een die grotendeels overeenkomt met HET GROENE BOEKJE (ruim 100.000 woorden). Bij het ontwikkelen van cursusmateriaal gebruiken we echter meestal een zelf samengestelde lijst van ruim 30.000 woorden, die slechts gedeeltelijk overlapt met de groene lijst. Er zijn dus net zoveel boekjes als kleuren.
> In TEXEDIT kan men real-time een tekst op de juiste spelling controleren. De juist gespelde woorden worden groen weergegeven, de foute woorden rood en woorden beneden een bepaalde omvang, bijvoorbeeld 4 karakters, wit. Het blijkt dat een op deze manier gekleurde file snel 'in orde' te brengen is, niet in de laatste plaats omdat accenten automatisch worden geplaatst. Bovendien worden bepaalde categorieën woorden, zoals passieve werkwoorden, geel of cyaan weergegeven, zodat men snel een 'beeld' van de stijl krijgt. Om dit alles te kunnen ondersteunen, moeten we kunnen beschikken over goede, al dan niet zelf samengestelde, lijsten van woorden."

Het WORDS-L team

Wat zou er nu meer voor de hand liggen dan een nieuwe, correcte woordenlijst te maken waar iedere tekstverwerker en iedere spellingchecker op ieder computersysteem in ieder land mee uit de voeten kan?

Deze vraag leidde tot het ontstaan van een WORDS-L-team (WLT) van mensen die vanuit hun woon- en werkplaatsen verspreid over Nederland werken aan een grote, correcte en bovenal kostenloze Nederlandse woordenlijst voor iedere gebruiker.

Eerst was het Gerard van Nes die de zomerse nacht voor zijn vertrek naar Florida offerde aan het programmeren in PERL van een utility, `match.pl` waarmee een bestaande woordenlijst kan worden geschoond en aangevuld volgens een aantal nieuwe spellingregels. Een slim PERL-script. Waar 'uiering' stond, ook als onderdeel van een woord, werd er het nieuwe 'uienring' van gemaakt. Wel moest het resultaat nog met de hand worden nabewerkt, want ook het woord 'sluiering' bijvoorbeeld werd voorzien van een extra letter. Zo bleken er meer *woordgrapjes* te zijn die moesten worden gecontroleerd. Gelukkig schreef het PERL-script van Gerard een verslagje van alles wat het had gedaan, zodat de correcties gemakkelijk konden worden uitgevoerd. Intussen was de discussie over woordenlijsten van TEX-NL verhuisd naar WORDS-L, een eigen mailinglijst vanaf FGBBS.

Van welke woordenlijst kon eigenlijk het beste worden uitgegaan? Het simpelst zou het natuurlijk zijn als iemand alle woorden uit het ELEKTRONISCH GROENE BOEKJE zou halen. Hans Linders van de TU EINDHOVEN liet zien dat dit zonder al te veel omhaal mogelijk is, maar de woordenlijst die hieruit rolde kan niet zomaar worden verspreid. Het was dankzij Linders wel mogelijk om duizenden woorden in HET GROENE BOEKJE op te zoeken zonder dat er dagen voor moest worden gebladerd of gewacht.

Copyright?

Op het Internet zijn hier en daar Nederlandse woordenlijsten te vinden, zelfs exemplaren van HET GROENE BOEKJE uit 1954 en 1990 (het eerste is nog 'gered' uit de stroken geponst papier waarop destijds de data werden bewaard), bijvoorbeeld op `gopher://olt.et.tudelft.nl:72/1/words/` van de TU DELFT en bij `ftp://ftp.nl.net/pub/textproc/dictionaries/` van NLNET.

De vraag is echter of die woordenlijsten 'vrij' zijn. Erick Branderhorst, lid van het WORDS-L-team hierover:

> "Internet en free software zijn onlosmakelijk met elkaar verbonden. Velen zijn bekend met bijvoorbeeld de GNU, XFREE, TEX en talloze andere soortgelijke projecten. Het vrij ter beschikking stellen van software houdt echter niet op bij het aanbieden van een bestandje op een server en het in een Nieuwsgroep bekend maken van dit bestandje. Het is wel degelijk belangrijk dat er duidelijk bij vermeld wordt wie het wel en wie het niet mag gebruiken en onder welke voorwaarden.
> Van alle vrij beschikbaar gestelde lijsten van Nederlandse woorden hebben we geen enkele lijst kunnen vinden waarvoor dit duidelijk was omschreven. De herkomst van de lijsten is vaak duister en niemand heeft klaarblijkelijk de moeite genomen om voorwaarden voor het gebruik afdoende te definiren. Wij willen voor eens

> en voor altijd een einde aan deze situatie maken en een pakket met Nederlandse woordenlijsten, afbreekpatronen en aanverwante zaken vormen en dit distribueren onder de voorwaarden zoals gesteld in de GENERAL PUBLIC LICENSE II (afgekort als GPL, er is ook een Nederlandse versie). Vrij vertaald behelst de GPL het volgende: je mag alles doen met dit pakket wat je wilt, als je er maar voor zorgt dat je alles wat jij kreeg ook weer beschikbaar stelt aan anderen."

We kozen voor het gebruik van een woordenlijst die niet van de redactie van HET GROENE BOEKJE afkomstig was. Erik Frambach had al eens een Nederlandse woordenlijst gecompileerd voor het gebruik bij AMSpell. Het script van Gerard van Nes genereerde een nieuwe lijst van 220.000 woorden die we `nl963.txt` hebben genoemd.

ISO LATIN1 & Piet Tutelaers

Vanaf dit punt werd Piet Tutelaers de spil waarom het project draaide. Aanvankelijk om de nodige afbreekpatronen te genereren, maar ook om de kwaliteit van de verkregen woordenlijst te verbeteren. Het was zijn idee om te werken aan een universeel masterbestand in het ISO LATIN1 formaat. Van daaruit kan voor elke denkbare karakterset een vertaling worden gemaakt, dus ook naar ASCII. Voor deze vertaalslag leverde Piet het PERL-script `lat2ansi.pl`

Piet schreef tijdens het werk aan de lijst het volgende:

> "Het bestand `nl963.txt` heb ik omgezet naar ISO LATIN1 en vervolgens geverifieerd met mijn benadering van het ELEKTRONISCH GROENE BOEKJE (EGB96). In mijn EGB96 benadering (laten we dit gemakshalve maar even E-G-B96 dopen omdat hierin ook de afbreekplaatsen staan) ontbreken werkwoordsvormen als werkt en werk-ten, echter niet werk-te en ge-werkt. Ook meervoudsvormen als kaart-jes (wel kaart-je) en de overtreffende trappen kaler, kaalst (wel kaal). Ik heb geprobeerd op basis van eenvoudige heuristieken de woordenlijst te screenen met mijn PERL-script `verifieer`. Het resultaat is `spell-nl-iso.txt`[2] Hierin worden de woorden voorafgegaan door codes die aangeven wat er nog met een bepaald woord moet gebeuren:

```
=<woord>
   <woord> identiek aan E-G-B96 (zonder
           afbreekstreepjes)
-<woord>:<woord1>
   <woord> moet vervangen worden
           door <woord1>.
-<woord>:<woord1>|<woord2>
   <woord> moet vervangen worden door
           <woord1> en/of <woord2>
?<woord>
  <woord> met twijfelachtige spelling.
          Bijvoorbeeld 'kapotmaakt'
          omdat dit los wordt
          geschreven 'kapot maakt'.
#<woord>
  <woord> ontbreekt in E-G-B96
          terwijl verwante
          woordvormen wel voorkomen
          (werkten, kaalst, etc.)
+<woord>
  <woord> ontbreekt in E-G-B96.
          Het lijkt mij verstandig
          deze woorden na te lopen
          in de Dikke van Dale.
```

> Mijn ideaal zou zijn een groot ISO-LATIN1 woordenbestand met correcte woorden, dus inclusief hoofdletters, koppeltekens (streepje), apostrof en *met* afbreekplaatsen (klein streepje).
> Dit bestand zou diverse behoeften kunnen dekken, zoals het genereren van afbreekpatronen voor TEX en spellingscontrole. Onze eerste taak is echter te zorgen voor een nauwkeurige woordenlijst. Hoe pakken we dat aan? Van de Dikke Van Dale bestaat een CD-ROM versie. Kunnen we een programma maken dat onze woorden uit de categorie + hiermee vergelijkt? Of moeten we dit zelf gaan doen? De categorie woorden met # en ? wil ik wel voor mijn rekening nemen evenals het toevoegen van de afbreekstreepjes. Ook moeten we ons afvragen of 'werkten' naast 'werkte' wel nodig is."

Het onvermijdelijke handwerk

Een aantal van deze vragen is nog onbeantwoord. Leden van het WLT controleerden elk een deel van de ongeveer 50.000 woorden waarvan niet automatisch kon worden bepaald dat ze correct zijn. Inmiddels was het besef doorgedrongen dat je een woordenlijst niet 'even' maakt of corrigeert, ook al heb je computers tot je beschikking, plus slimme mensen om ze te bedienen...

Bij het nakijken van de 50.000 te controleren woorden was te zien dat ergens in de voorgeschiedenis ervan een programmeur bezig was geweest om woorden bij te maken. Er stonden woorden als 'complexmeest' en 'beduusdmeest'. Kennelijk had zijn programma de overtreffende trap in *complex, complexer, meest complex* verkeerd genterpreteerd. Piet Tutelaers wist de etymologie van deze fout te traceren:

> "Ongetwijfeld heeft iemand deze per computer gefabriceerd uit GB54, de woordenlijst van HET GROENE BOEKJE uit 1954. Hierin komen woorden voor als:
> `110000000#bed*uusd#bed*uusde#bed*`
> `uusder#meest bed*uusd#06`
> De superlatief van beduusd is dus niet beduusdst maar 'meest beduusd'. Alleen welke snoodaard heeft dit omgedraaid en de spatie weggelaten?"

We zullen het nooit weten, maar het betekende wel dat we alle woorden eindigend op "meest" konden verwijderen (behalve "allermeest").

In de datafile van HET GROENE BOEKJE uit 1954 stond trouwens

```
910000000#boeddh*istisch#boeddh*istische#
boeddh*istischer#boeddh*istischt#76
```

Hieruit blijkt dat het woord 'boeddhistischt' destijds door de beugel kon. Onzin natuurlijk en in HET GROENE BOEKJE van vandaag staat dat ook niet meer. Woorden als deze maakten het des te verstandiger dat we een aantal woorden met de hand naliepen of, beter gezegd, met de hand aan het toetsenbord en met de woordenboeken naast het scherm.

Het woordenlijst-checken (Jacoline en ik 'deden' meerdere happen van ongeveer 6000 woorden per keer) werkte

[2] Inmiddels zijn er nieuwe versies verschenen, zie de verwijzingen onderaan dit artikel.

bij ons op een bepaalde manier verslavend. Na een uurtje merkten we dat we eindelijk de kop eraf hadden, aan het schuifblokje dat rechts in het scherm van de editor aangeeft waar je ongeveer zit in het hele bestand. De eerste tijd, als het blokje aan het begin blijft staan net alsof er nog *niets* is gepresteerd, is het moeilijkst. Daarna was het zaak af en toe te pauzeren en vooral niet door te willen prakken als de letters gaan dansen en we niet meer lekker op de stoelen konden zitten. Vervolgens, wanneer we eenmaal over de helft heen waren, leek het wel alsof de zwaartekracht meehielp en zakten we in een paar uur (koffie en thee tussendoor) naar het einde, tot het begeerde `<EOF>` teken ineens van onderaf het scherm opschoof...

Dwaas, dwazer, dwaast?

We ontdekten overigens ook dwaasheden in de zogenaamde 'standaardwerken' waarin we doorlopend naar wijsheid zochten bij twijfelgevallen. Onze Dikke van Dale gaf nota bene aan dat je *onzichtbaar, onzichtbaarder, onzichtbaarst* kunt schrijven! Belachelijk. Je bent zichtbaar, slecht zichtbaar of onzichtbaar, maar als niemand je meer kan zien, kun je niet nog wat onzichtbaarder worden, toch? De Dikke kan dunner, is onze conclusie.

Zelf hebben we vast en zeker nog wat fouten laten zitten. Een eerste, van mijn eigen hand, is onlangs ontdekt: ik had het woord 'aangeft' laten staan wat onzin is. De trema moet eraf!

Helemaal 'af' zal de woordenlijst niet komen, want de Nederlandse taal wordt gemaakt door de Nederlanders. En die raken nooit uitgepraat. Dit wordt prachtig gellustreerd door Wim Danils en Felix van de Laar, de auteurs van het boek SPELLINGCHAOS. Zij schreven een verslag van hun naspeuringen naar de verwikkelingen die hebben geleid tot de nieuwe spelling (de officile WOORDENLIJST). Het is ronduit onthutsend maar vaak ook vermakelijk om te lezen hoe de diverse hotemetoten van gewichtige instituten langs elkaar heen blunderen. Het blijkt dat de hoogste instanties op het gebied van de nieuwe spelling het op belangrijke punten met elkaar oneens zijn. In plaats van elkaars kritiek te verwerken in verbeteringen, bestrijdt men de kritiek en dat levert tot standaard verheven dwaasheden op (bij het schrijven van het volgende heb ik gegevens en zinnen ontleend aan het boek van Danils en Van de Laar):

- in de WOORDENLIJST zou de *dubbelspelling* nu verdwenen moeten zijn, zodat alleen *akkoord* nog goed is, en *accoord* niet meer. Maar: msli is gebleven naast muesli, shockeren naast choqueren, clerus naast klerikaal, vacant naast vakantie, sex-appeal naast seksshop en ga zo maar door.
- een nieuwe spellingregel zegt dat namen van tijdperken met een hoofdletter moeten, maar dat wordt officieel vergeten bij de oudheid, de bronstijd, de ijstijd en de steentijd.
- willekeur: bij de nieuwe spellingregels staat dat het woord *pistool* moet worden afgebroken als pi-stool, maar het is wel waterpis-tool, laserpis-tool en laspis-tool!
  Een stereo-installatie heeft een streepje na stereo, maar videoinstallatie niet.
- de officile WOORDENLIJST bevat tientallen spel- zet en/of drukfouten. Bij herdrukken van HET GROENE BOEKJE is een aantal daarvan stilzwijgend verbeterd en ook is er op een gegeven moment een velletje met *errata* bijgevoegd, maar nog altijd blijven er veel bekende fouten zitten (lijkt dit niet erg op de eerder gemelde houding van WP?).
- hoofdletters: sommige volkeren verdienen er wel een (de Batavier, de Saks) maar andere niet (de azteek, de zoeloe en de bosjesman).
  Bij de ziektes moet pfeiffer het zonder hoofdletter stellen, Alzheimer heeft hem wel gekregen. Het lijkt wel een lintjesregen! De oscar, bekende filmprijs, heeft geen hoofdletter, tenzij hij wordt uitgereikt: Oscaruitreiking.
- de tussen-n zorgt voor veel hilariteit. 'Pannenkoek' is de beroemdste geworden, maar het is erg gesteld met de belastingen. 'Aangiftebiljet' hoort zonder tussen-n maar hij moet wel staan in 'belastingaangiftenbiljet'.
  Oorzaak is dat de samensteller van de officile woordenlijst — INSTITUUT VOOR NEDERLANDSE LEXICOLOGIE, INL samen met de TAALUNIE — is uitgegaan van een verzameling documenten die eerst als standaard is aangenomen en waaruit vervolgens pas de woorden zijn gesorteerd. De samensteller nam zich voor onder geen beding af te wijken van deze standaard-input en verdedigt deze dan ook te vuur en te zwaard tegen elke inmenging van buitenaf... *tenzij* de samensteller vindt dat er nog iets moet worden veranderd, bijvoorbeeld toen deze ontdekte dat wel het woord *bondsdagpresidente* was opgenomen, maar niet het woord *bondsdagpresident*...

Het stof, dat de introductie van de nieuwe spelling deed opwaaien, is nog niet gaan liggen. Het ANP hanteert de nieuwe spelling zoveel mogelijk, voor zover als ze het tenminste kunnen bijhouden. De Vlaamse krant DE STANDAARD vaart zijn eigen koers en geeft daarmee de toon aan voor veel andere kranten. NRC HANDELSBLAD heeft besloten het woord *product* te blijven spellen als *produkt*.

Slotsom

Mijn conclusie: nu de oude spelling wel door iedereen van gezag is verlaten, zijn er vele, soms kostbare en tijdrovende publicaties te koop die beloven een leidraad te geven. Geen enkel bedrijf biedt een leesbare en betrouwbare woordenlijst van enige omvang aan waarmee de gebruiker zelf een spellingchecker kan uitrusten. Het WORDS-L-team is bezig hiervoor een uitstekend alternatief aan te bieden. Inmiddels is de WORDS-L-mailinglijst[3] vervangen

[3]Om een indruk te geven van het drukke en dagelijkse overleg: vanaf 11 juli tot 14 september werden er in WORDS-L 386 berichten geschreven.

door een sneller alternatief bij de TU EINDHOVEN. Door een spelfout (!) is deze per ongeluk SPELING[4] gaan heten.

De bestanden en programma's die we gebruiken, alsmede de resulterende woordenlijsten, zijn te vinden op het Internet en op FGBBS. Gebruikers van *middle-of-the-road* tekstverwerkers als WP en MS-WORD willen we zelfs zoveel mogelijk tegemoet komen door het beschikbaar stellen van plug-and-play modules waarin onze lijst is verwerkt (bij de laatste versie van WP lijkt dat moeilijker en voor MS-WORD het lastigste, maar Hans Linders is hiervoor aan het werk gegaan). Ook is een van de leden van het WLT begonnen met het programmeren van een spellingchecker die nauwer samen zou moeten gaan werken met verschillende tekstverwerkers.

Diverse referenties op het Internet

Let wel: deze adressen en filenamen zijn in ontwikkeling, en daardoor aan verandering onderhevig.

- `http://www.pi.net/~fg/words.htm`
  HTML-versie van dit artikel, met vele links naar relevante sites
- `ftp://ftp.tue.nl/pub/tex/GB95/`
  Bestanden van het WORDS-L project
- `ftp://ftp.iaehv.nl/pub/users/branderh/words-l/`
  Een *nix verzamelpackage, neergezet door Erick Branderhorst (`branderh@debian.iaehv.nl`)
- `ftp://ftp.tue.nl/pub/tex/GB95/spell-nl`
  SPELL-NL bestanden van het WORDS-L project
  In deze directory staan:
  - `ftp://ftp.tue.nl/pub/tex/GB95/spell-nl/LEESMIJ`
    LEESMIJ, een toelichting
  - `ftp://ftp.tue.nl/pub/tex/GB95/spell-nl/COPYING.nl`
    COPYING.nl , over copyrights
  - `ftp://ftp.tue.nl/pub/tex/GB95/spell-nl/spell-nl-ansi.v3`
    spell-nl-ansi.v3, woordenlijst versie 3, ANSI formaat (voor DOS / OS/2, WIN PC's)
  - `ftp://ftp.tue.nl/pub/tex/GB95/spell-nl/spell-nl-ansi.zip`
    ZIP versie spell-nl-ansi.v3 , woordenlijst versie 3, ANSI formaat (voor DOS / OS/2, WIN PC's)
  - `ftp://ftp.tue.nl/pub/tex/GB95/spell-nl/spell-nl-iso.v3`
    spell-nl-iso.v3 , woordenlijst versie 3, ISO-LATIN formaat
  - `ftp://ftp.tue.nl/pub/tex/GB95/spell-nl/spell-nl-iso.zip`
    ZIP versie spell-nl-iso.v3 , woordenlijst versie 3, ISO-LATIN formaat
  - `http://net.info.nl/ci/`
    Computer Info
  - `ftp://ftp.iaehv.nl/pub/users/branderh/words-l/gpl-nl.html`
    html versie van de nederlandse public license

Enkele schriftelijke publicaties

- SPELLINGCHAOS, door Wim Danils en Felix van de Laar, Uitgeverij Scheffers, Utrecht, 1996, ISBN 90.5546.048.6
- WAT VERANDERT ER NU EIGENLIJK?, door Nel Korstanje en Jan Heerze, Walvaboek, Laren Gld, 1995, ISBN 90.6675.589 X
- VAN DALE EN DE NIEUWE SPELLING, van Dale Lexicografie, Utrecht / Antwerpen 1996, ISBN 90.6648.998.7 (met een beetje geluk gratis *ritselen* bij de betere boekhandel)
- SPELLINGBESLUIT, SDU uitgevers, Den Haag en Standaard Uitgeverij, Antwerpen, 1996 (regels en bijna 16.000 probleemwoorden, in de aanbieding voor fl 9,90)

[4] 153 berichten in de eerste maand.

Heeft TEX nog toekomst?

Hans Hagen

pragma@pi.net

September 1996

1 Inleiding

Dat TEX in zijn pure vorm geen eenvoudig systeem is, zal door velen volmondig worden bevestigd. Soms wordt binnen de TEX-wereld met trots verwezen naar al die organisaties die deze taal en het gelijknamige programma gebruiken. Maar hoeveel mensen hebben na een eerste, wellicht te vluchtige blik, al niet TEX de rug toegekeerd?[1] Niet zelden maken gebruikers melding van een moeizaam implementatieproces, waarin de nodige hobbels moesten worden genomen.

Zowel de complexiteit van als de bewondering voor TEX hebben ertoe geleid dat in veel landen gebruikersverenigingen zijn opgericht. Niet zelden wist men van de nood een deugd te maken, zodat naast de nodige netwerken ook de noodzakelijke expertises zijn opgebouwd. Zo ook in Nederland, waar de $\mathcal{NTG}$ zich mag verheugen in een constante belangstelling en een immer gevuld verenigingsblad. Het kan geen kwaad na vijftien jaar TEX en bijna tien jaar $\mathcal{NTG}$ eens terug te kijken wat er is bereikt en vooruit te kijken naar wat er gaat komen.

Ik zal hieronder een aantal observaties beschrijven, zonder daarbij de pretentie te hebben volledig te zijn of alles feitelijk juist weer te geven. Het betoog mondt uit in een aantal vragen die zowel binnen het bestuur van de $\mathcal{NTG}$ als bij de leden de nodige aandacht verdienen.

2 Standaard

De meesten van ons werken op dit moment met TEX versie 3.14159. Voor alle duidelijkheid: we hebben het hier over de macrotaal en het programma. Vrijwel iedereen gebruikt wel een macropakket dat de gebruiker vrijwaart van al te veel programmeerwerk. Van alle ooit geschreven macropakketten wordt LATEX het meest gebruikt. In die zin is voor veel mensen TEX synoniem aan LATEX.

Uit het eenvoudige gegeven, dat elke nieuwe versie van de macrotaal en het programma TEX het versienummer metéén decimaal in nauwkeurigheid doet toenemen, kan men afleiden dat we te maken hebben met een uitermate stabiel systeem. Zelfs de sporadische updating van TEX is schijn. De functionaliteit is immers bevroren en de officile versie-aanduiding is dan ook TEX82. Het langzaam naderen tot π heeft, afgezien van het ondersteunen van meerdere talen in een format en de voor velen onbekende virtuele fonts, dan ook vooral betrekking op het wegwerken van bugs.

Er zijn binnen de TEX-wereld verschillende commissies werkzaam, bijvoorbeeld op het gebied van het vastleggen van het DVI formaat, specials en de structuur van directories. Knuth ging er, toen hij TEX ontwierp, vanuit dat het systeem minstens honderd jaar mee ging. Het feit dat de huidige standaard van het DVI formaat, waarvoor een commissie in het leven is geroepen, nauwelijks afwijkt van die van TEX82, is in die zin een goed teken.[2] Is het aan de andere kant niet verbazingwekkend dat na vijftien jaar er nog geen standaard is geaccepteerd voor specials en dat de directory structuur nog deels experimenteel is? En dat terwijl het aantal macropaketten, en dus de gewenste functionaliteit, nu niet bepaald de pan uit rijst.[3]

> Het is blijkbaar eenvoudiger een moeilijke ISO standaard te ontwikkelen dan een eenvoudige TEX standaard.

Er is een terrein waar TEX voortdurend in ontwikkeling is, namelijk dat van de fonts. Al sinds het ontstaan van dit blad kunnen we in de TUGBOAT indrukwekkende artikelen lezen over de wijze waarop de verschillende mensen TEX geschikt maken om ook in hun taal en schrift zetwerk te leveren.

> De meest indrukwekkende vooruitgang wordt binnen de TEX-wereld geboekt op het gebied van fonts.

Hoewel voor velen TEX nauw samenhangt met esthetica, zijn het met name de fonts, en dus METAFONT, waar dit aspect het meest tot zijn recht komt.

We mogen in dit verband niet onvermeld laten dat juist fonts de meest problematische factor zijn bij het inrichten van een robuuste TEX werkomgeving, hoewel de ontwikkelaars van viewers het ons gelukkig steeds gemakkelijker maken.

---

[1] Tenzij uit de context anders blijkt, kan men voor TEX ook de naam van een daarop gebaseerd macropakket lezen.

[2] Recente experimenten met de opcodes *xxx*2 en *xxx*3 hebben ondergetekende geleerd dat bouwers van DVI drivers het op dat punt niet zo nauw nemen met de standaard en daarmee een van de kanalen van TEX naar buiten hebben geblokkeerd.

[3] Aan de zijlijn wordt door een kleine groep gebruikers het probleem rond de specials op een andere manier aangepakt. In die zin lijkt er nog wat moois aan te komen.

3 Werkomgeving

De $\mathcal{NTG}$ mag gepast trots zijn op een tot voor kort uniek initiatief, namelijk het inrichten van een CDROM, met daarop al het goede dat de TEX-wereld te bieden heeft: 4TEX. Het enthousiasme waarmee dit plug-and-play produkt wordt onthaald, is een steun in de rug voor TEX-gebruikers. Wat wellicht menig gebruiker ontgaat, is dat achter die inmiddels meer dan 1000 megabytes aan sources, fonts, style-files, documentatie en hulpprogramma's een wereld van mogelijke verwarring schuil gaat. De charme van TEX was immers dat men met enkele megabytes de meest fantastische resultaten kon bereiken. Nu wil ik niet beweren dat iedere gebruiker precies wist wat er gebeurde, maar een basaal inzicht in ASCII, TEX, `tfm`, `pk`, DVI en dergelijke is nog wel te verwerven.

Hoewel een vorm van abstractie de drempel kan en ongetwijfeld zal verlagen, bestaat op termijn de kans dat de gebruiker door de bomen het bos niet meer ziet. Persoonlijk heb ik het al opgegeven ruwe TEX files te downloaden. Ze zijn namelijk niet te verwerken als je niet beschikt over de juiste macro's, die soms weer afhangen van dialecten, er niet bij hebt. Met DVI files zal het, vrees ik, dezelfde kant op gaan. Hoeveel, al dan niet interim, encoding vectoren worden er inmiddels gehanteerd?

> De ooit zo onafhankelijke TEX gebruiker onderscheidt zich steeds minder van gebruikers van andere grote systemen.

Enerzijds kunnen we ons nieuwsgierig afvragen waarom het zo lang heeft geduurd voordat een suite het leven zag, anderzijds kunnen we ons verontrust achter de oren krabben als we ons realiseren of we zijn opgegaan in de whatever-office wereld. Een voordeel maar mogelijk ook nadeel is dat je in die wereld extra kritisch wordt bekeken en vergeleken en zo slachtoffer kunt worden van je eigen succes. Hoe dan ook:

> Grootschalig gebruik van TEX vereist een ander soort ondersteuning.

4 Invalshoek

Men kan bij het opzetten van een tekst vertrekken vanuit de structuur, de inhoud, de opmaak of een combinatie hiervan. Een ieder leert tegenwoordig tijdens zijn opleiding wel op een of andere manier een tekst op gestructureerde wijze op te zetten. Het vertrekpunt wordt daarbij altijd gevonden in een soort raamwerk. Deze leerervaring wordt echter even zo snel weer negatief gecompenseerd door het leren werken met tekstverwerkende systemen die dwingen te denken in lettertypes, harde returns, tabs en andere eigenaardigheden. Geconfronteerd met een onbevredigende layout — als men daar tenminste oog voor heeft — ziet men zich gedwongen de nodige typografische correcties aan te brengen.

TEX maakt het daarentegen mogelijk een tekst vrijwel volledig te specificeren in termen van structuur. Deze sterke kant van TEX wordt vaak genoemd en geroemd maar heeft tot op heden nog niet geleid tot een samenhangend systeem dat structureel tekstverwerken tot in de puntjes ondersteunt. Natuurlijk ondersteunen de meeste macropakketten gestructureerd tekstverwerken, maar de toegenomen complexiteit en omvang van documenten vraagt soms om meer.

Ik ben jaloers op Knuth's gave om zijn gedachten op papier vast te leggen. Hoe briljant zijn boeken ook zijn, de beginnende gebruiker zal bij het lezen daarvan al snel de moed in de schoenen zinken. Even snel TEX of METAFONT leren is er niet bij. Gebruikers die bij het zetten van formules toch moeten afdalen naar de basis, kunnen gelukkig terugvallen op wat eenvoudiger, maar niet minder goed geschreven boeken als A Beginners Book of TEX (Levy & Seroul). Voor de programmeurs is er natuurlijk TEX by Topic (Eijkhout).

Eigenlijk houdt het daarmee op. TEX vormt een fantastisch startpunt voor het gedegen uitwerken en aanleren van gestructureerd tekstverwerken. De uitdaging die Knuth in de ring heeft gegooid met zijn collegediktaat Mathematical Writing is voor zover ik weet nooit opgepakt.[4] Wellicht heeft het besef dat ik voor dat unieke maar toch wel pittige college slechts met moeite zou zijn geslaagd — elke keer als ik het diktaat doorkijk realiseer ik me dat ik nooit vergelijkbare colleges heb gehad — ertoe geleid dat ik nog nooit verder ben gekomen dan het schrijven van drie hoofdstukken van een boek over gestructureerd tekstverwerken.

> Terwijl in opleidingen wordt gehamerd op structuur in teksten, is de bijdrage van TEX aan gestructureerd tekstverwerken minimaal geweest.

Misschien ligt de echte reden voor deze tijdelijke inzinking mede in het feit dat ik bij het uitwerken van dit onderwerp niet anders kan dan voorbeelden geven in een systeem dat geschreven is in TEX. En welke potentile lezer gebruikt zo'n systeem? Het feit dat de meeste TEX gebruikers gebonden zijn aan LATEX maakt de doelgroep er ook al niet groter op. Is het overigens niet tekenend dat de TEX-wereld wordt gedomineerd door n pakket, terwijl eigenlijk iedere categorie toepassingen zijn eigen aanpak vergt? Is dit niet volledig strijdig met de uitgangspunten van TEX?

Ik wil hier overigens niets afdoen aan de kwaliteit van LATEX. Men moet zich echter wel de vraag durven stellen of een wereldwijde bemoeienis met de ontwikkeling van zo'n pakket tot iets zal leiden. Waar eerst Lamport en later anderen als 'eenling' in staat waren hun visie te vertalen in een macropakket, moeten bij het verder uitontwikkelen vele tientallen (zo niet meer) visies tot een consistent geheel worden gemaakt. Dat dit maar matig is gelukt is te zien aan de bonte verzameling style files by LATEX. Wat voor veel wat meer onderlegde gebruikers wellicht een charme is, lijkt waarschijnlijk voor niet-technische gebruikers een chaos.

Dit brengt me op wat anders. TEX is vrijwel een synoniem voor wis- en natuurkundig zetwerk. Ondergetekende heeft daarentegen juist ondervonden hoe bruikbaar TEX is bij het

[4] Dit diktaat behandelt een mix van vormgeving, schrijfstijl en structuur.

manipuleren van teksten. Pas las ik in de beschrijving van een opmaaksysteem dat het mogelijk is tekst conditioneel te zetten. Een ander systeem maakte trots melding van de unieke mogelijkheid symbolische verwijzingen te hanteren. TEX kan dat al vijftien jaar!

Ik durf hier te beweren dat buiten de wiskundige wereld een veel grotere markt ligt voor TEX. Waarom wordt TEX niet massaal gebruikt als front-end voor database programma's? Is TEX niet bij uitstek geschikt voor de sociale wetenschappen, waar veel met teksten wordt gewerkt? Overigens werken veel sociale wetenschappers al decennia met statistische programma's die dicht tegen programmeren aanliggen en een lelijke output leveren. Een braakliggend terrein zou ik zeggen. En wat te denken van hen die dagelijks met de geschreven taal omgaan?

> De focus op wiskundig zetwerk heeft erin geresulteerd dat TEX niet die plaats heeft gekregen die ze verdient.

Overigens willen m'n collega en ik ooit nog eens een TEX voor kinderen ontwikkelen. We hebben de basis en de programma's per slot van rekening al klaarliggen. Jong geleerd is immers oud gedaan.

5 Ontwikkeling

Let wel, ik wil hier al die ontwikkelaars niet te kort doen. Er zijn fraaie en indrukwekkende dingen bereikt, neem alleen al de ondersteuning van allerlei oosterse talen. En wie had ooit gedacht dat met TEX muziek kon worden gezet? Zelf werken wij naar volle tevredenheid met het rond 1988 door Wichura ontwikkelde macropakket TABLE, dat enerzijds een schoolvoorbeeld is van een stabiel, af en goed pakket, maar anderzijds een evenzo groot voorbeeld is van miskend werk.

TEX kan overigens niet alleen gebruikt worden om gestructureerd te leren tekstverwerken. TEX kan namelijk ook een zeer motiverende inleiding in programmeren zijn, en zeker in recursief denken. Welke andere taal geeft zulke intrigerende output? Helaas is wat ik gemakshalve altijd maar aanduid als typografisch programmeren nog nooit echt van de grond gekomen. Het lijkt me een uitdaging zo'n vak te geven.

Het is in dat verband aardig stil te staan bij recente ontwikkelingen. De komst van PDF maakt het mogelijk teksten te voorzien van geavanceerde zoekstructuren. Er onstaat een nieuw soort teksten met verrassend nieuwe mogelijkheden. Waar vroeger de auteur en/of vormgever afhankelijk waren van de esthetische kwaliteiten van de programmeur — "Sorry, maar wat u wilt is technisch echt onmogelijk.-– wordt hij nu zelf programmeur.

Ruim een jaar geleden heb ik in TEX een in mijn ogen visueel aantrekkelijke omnummergids geschreven. Zoiets kost ruim een middag werk, waarbij de meeste tijd gaat zitten in het uitproberen van verschillende layouts.[5] Het aardige van deze demo is dat de naeve gebruiker de indruk heeft te werken met een programma, terwijl het in feite een passieve tekst betreft. De grenzen tussen vormgeven en programmeren vervagen en een hele reeks aan nieuwe toepassingen ligt in het verschiet. Inderdaad:

> TEX is nog steeds de enige echte typografische programmertaal.

In de toekomst zullen we worden geconfronteerd met digitaal papier. Lezen van een scherm, of wat daarvoor in de plaats moet komen, wordt vanzelfsprekend. In mijn ogen is TEX op dit moment het systeem bij uitstek om de grenzen van dit nieuwe medium te verkennen, net zoals TEX het eerste systeem was dat geavanceerde PDF documenten mogelijk maakte.

6 TEX Valley

Nu we het toch hebben over talen, kunnen we meteen wel constateren dat er een verschuiving optreedt in het zwaartepunt van de ontwikkeling rond TEX. Het begon in de States, maar momenteel lijkt de TUG een kwijnend bestaan te leiden: mijn rijtje MAPS haalt het rijtje TUGBOAT langzaam in. Op dit moment gebeurt er veel in Europa, zij het dat de innovaties meer en meer in het oosten van dit contingent plaatsvinden. Het kan dan ook bijna niet anders dan dat de komende tijd het verre oosten de kar gaat trekken. Zolang zij niet worden ondersteund door andere programma's, zal TEX daar ht hulpmiddel worden!

> Blijkbaar kan men een beperkte tijd pieken, om dan te verworden tot een bezadigde TEX gebruiker.

Of ligt het wat anders? Voordat wij min of meer noodgedwongen besloten zelf een macropakket te schrijven hebben wij eerst rondgekeken. Naast het op dat moment erg Amerikaanse en in functionaliteit beperkte LATEX waren er $\mathcal{AMS}$-TEX, L$\mathcal{AMS}$-TEX, INRSTEX en nog wat initiatieven. We hebben ze allemaal uitgeprobeerd. Waarom heeft die prettige variatie in pakketten niet doorgezet? Heeft niet iedere dicipline behoefte aan een gerichte, gentegreerde oplossing? Ik heb in een internet discussie eens iemand horen stellen dat de TEX-wereld uitblinkt in het navelstaren. Net nieuw op het net, meende ik dat te moeten weerleggen. Zou ik dat nu nog doen?

7 Navelstaren

Het is bij een gecompliceerde tekst niet altijd even eenvoudig elk specifiek element in abstracties te definiren. TEX valt onder de categorie 'intentionele zetsystemen', maar in veel gevallen kan de gebruiker zijn intenties niet kwijt. Niet zelden zie je in sources die op het net circuleren dat intenties zijn verworden tot in-line TEX. Wellicht de meest voor de hand liggende oorzaak hiervan is dat de gebruiker gebruik maakt van TEX omdat er simpelweg geen ander systeem voorhanden is. TEX is in zo'n geval niet meer dan een gewone doorsnee tekstverwerker. Als daarnaast niet ook de nodige zorg is besteed aan het wiskundig zetwerk, dan valt ook dat voordeel weg.

[5] Zo kostte het maken van een interactief RGB-CMYK kleurenpallet niet veel meer tijd. Beide demo's zijn voor liefhebbers beschikbaar.

Helaas hebben de macropakketten die rond TEX zijn ontwikkeld hier een stevige bijdrage aan geleverd. Hoeveel mensen zijn niet gedwongen te denken in termen van sections en subsections, terwijl ze veel liever zouden spreken van procedures en werkinstructies? Nu hoor ik de wat meer ervaren TEX gebruiker al zeggen dat men eenvoudig een en ander kan herdefiniren, maar ik kan deze oplettende lezer verzekeren dat er situaties zijn waarin herdefiniren vrij lastig, zo niet onmogelijk is. Als we bijvoorbeeld naast hoofdstukken op hetzelfde niveau werkinstructies hebben, maar een andere vormgeving wensen en andere (lokale) lijsten willen genereren, dan komt daar bij de meeste macropakketten het nodige hackwerk bij kijken. En dit is slechts een eenvoudig voorbeeld.

> Specifieke beroepsgroepen en werkterreinen lijken nauwelijks te worden ondersteund.

Waar Knuth ons de tools gaf om dat te maken wat de situatie vroeg, heeft een ongewenste versmalling plaatsgevonden. Deze versmalling beperkt niet alleen het denken in termen van structuur, maar heeft bovendien een verlammend effect. Een voorsprong van vijftien jaar dreigt de komende jaren te worden omgezet in een achterstand. Het enthousiasme waarmee het grote publiek de plotsklaps cryptische (pseudo) structurerende opmaakcommando's in HTML omarmt en misbruikt (waar het om structuur gaat) is daarvan een veeg voorteken. De laatste jaren, of misschien zelfs pas het laatste jaar, is het plotseling n om te denken in commando's en ASCII. Hoe lang is het ook al weer geleden dat de tekst formatters op main-frames en de bij zetters gebruikte commando georinteerde systemen met alle geweld plaats moesten maken voor What You See Is What You Get? Voor al die nieuwe internetters gaat blijkbaar een wereld open. Het is de vraag of de zeer ervaren TEX-wereld de chaos die dreigt te ontstaan nog een halt kan toeroepen.

8 Accenten

Ik heb er de MAPS en de TUGBOAT niet op gecontrolleerd, maar het aantal artikelen over structuur, onderhoudbaarheid, efficincy en eenvoud valt in het niet bij het aantal dat macro's behandelt. Blijkbaar lopen auteurs hier niet warm voor.

> Het lijkt erop dat de meeste TEX gebruikers denken in macro's.

Natuurlijk staan er regelmatig bespiegelingen in en worden ontwikkelingen beschreven. Persoonlijk vind ik het altijd weer leuk als een MAPS of een TUGBOAT in de bus ligt.

Het is in dat kader de vraag op welke doelgroep de gebruikersgroepen mikken. Misschien is de groep te karakteriseren als systeembeheerders. Willen de eindgebruikers, die het liefst helemaal niet worden lastig gevallen met definities, installatie en andere technische zaken, niet liever weten he je TEX kan inzetten? Op welke manier iets slimmer kan? Waar winst te behalen valt? Hoe je iets vormgeeft? Wat gestructureerd tekstverwerken eigenlijk is?

Hier speelt een dilemma. Zowel de ervaren gebruiker als de meer technisch ingestelde beginner willen natuurlijk wel degelijk lezen wat op dit moment wordt aangeboden. Voor velen is TEX immers tevens een hobby. Het eindeloos steeds weer uitleggen aan het net nieuwe niet-technische lid wat TEX is en hoe dit of dat werkt, kan hem nauwelijks boeien. Hij wil nieuws!

> Kunnen zowel TEXneuten als eindgebruikers worden bediend door n gebruikersgroep?

of misschien:

> Zou eigenlijk niet iedere generatie gebruikers zijn eigen gebruikersgroep moeten hebben?

Dit is een essentiële vraag. Ikzelf zou een niet-technische gebruiker niet zo snel op een gebruikersgroep huidige stijl afsturen. Ik kan me rond een macropakket zeer gerichte activiteiten voorstellen, bijvoorbeeld sessies waarin voorbeelden worden uitgewerkt, vragen worden beantwoord, concepten worden uitgelegd, geleerd wordt hoe men het onderste uit de kan haalt, duidelijk wordt gemaakt op welke wijze een tekst moet worden opgezet, enz. De terugkoppeling die tijdens dergelijke bijeenkomsten ongetwijfeld plaatsvindt, kan vervolgens weer worden vertaald in verbeteringen en innovaties.

9 Toekomst

Heeft TEX een toekomst? Deze vraag kan bevestigend worden beantwoord, zolang tenminste de softwareontwikkelaars nog niet het inzicht hebben gehad de sterke kanten van TEX te omarmen en op te nemen in hun tekstverwerkers. Deze blindheid is overigens opmerkelijk als men zich realiseert dat de goudader aan de oppervlakte ligt: TEX is gepubliceerd! We kunnen dan ook rustig het volgende constateren.

> De eigenwijsheid van de gemiddelde programmeur en zijn denken het wiel nog ronder te kunnen uitvinden dan een ander, is de beste bescherming die TEX heeft.

De TEX-gebruiker heeft zich dan ook de afgelopen jaren in de sjieke omstandigheid bevonden in het verlengde van eeuwen typografische traditie beduidend meer dan gemiddeld zetwerk te kunnen opleveren. Een kwalitatief en conceptueel sterke concurrent zal het verdedigen van TEX nog moeilijker maken dan nu al vaak het geval is.

Zo nu en dan komt binnen de $\mathcal{NTG}$ weer eens een van de doelstellingen uit de wandelgangen naar boven: namelijk aansluiting op SGML. Dit toverwoord is synoniem aan gestructureerde opslag van tekstuele informatie. Met het grootste gemak worden tegenwoordig bestaande tekstverwerkers voorzien van een SGML interface. Bij sommige systemen werkt dit ietwat op de lachspieren, immers: hoe kan men teksten die opmaak en niet structuur als vertrekpunt hebben converteren naar structuur? Zou TEX op dit vlak misschien een leidende rol kunnen spelen?

Ik herinner me een congres in 1993 rond SGML waarbij de proceedings zowel op papier als elektronisch zouden worden opgeleverd. De papieren versie heb ik vier maanden na dato ontvangen, de elektronische versie is vrees ik

nog steeds in ontwikkeling. Ik zal de details achterwege laten, maar het boekwerkje van 75 bladzijden met een inhoudsopgave en twee registers was een tussendoortje geweest voor TEX. Uit de begeleidende brief werd wel duidelijk welke worsteling de vertaalslag van word-processor naar SGML naar gezette tekst heeft opgeleverd. Voor hen die TEX kennen voorwaar een sneue vertoning, waaraan bovendien alle vooraanstaande leveranciers van tekstverwerkers en een database leverancier hadden meegewerkt. Ja, hier had TEX inderdaad een leidende rol kunnen spelen.

Zolang men ergens een conversieprogramma op los kan laten, kan men van het ene naar het andere systeem overgaan.[6] Wat dat betreft biedt SGML als enige voordeel dat het een gestandaardiseerde wijze van vastleggen is. De belangrijkste randvoorwaarden zijn echter structuur en functionaliteit. Dat wil zeggen dat tekst niet alleen structuur moet bevatten, maar dat ook alle noodzakelijke informatie in de tekst besloten moet liggen. Wat er niet in zit, kan er immers ook niet uitkomen.

Nu wil het geval dat binnen de TEX-wereld men zich in hoge mate van beide randvoorwaarden bewust is, zij het dat ze niet altijd worden gehonoreerd. Het is echter de vraag of diezelfde TEX-wereld deze voorsprong weet om te zetten in een voorsprong op SGML gebied. Heeft zij zich ook al niet eens laten verrassen door HTML? We zeiden het al eerder: dezelfde gebruikers die eerst ASCII verfoeiden en TEX commando's niet konden aanzien, verlustigen zich nu in vishaakjes, `url`'s, mime types en onleesbare sources.

> *Alle pogingen om TEX onder het volk te brengen hebben jammerlijk gefaald, maar: ingewikkeld is n, nu TEX nog!*

Trouwens, hebben we ons al niet het gras voor de voeten laten wegmaaien? Waar DVI al jaren op alle platforms te bekijken is — we vergeten voor het gemak even alle encoding problemen — gaat PDF het roer overnemen. De interactieve meerwaarde is daarbij slechts schijn. Ook dat kon al jaren in DVI, dankzij de experimenteerdrift van de ontwikkelaars van drivers.

Wij werken nu zo'n zes jaar intensief met TEX, waarvan de laatste vier jaren met ConTEXt. Deze periode is een voortdurende zoektocht geweest naar de grenzen van TEX. Voor zover ik daar natuurlijk zicht op heb, is het einde van deze tocht nog niet in zicht. Zo was TEX bijvoorbeeld het eerste systeem waarmee zeer geavanceerde interactieve documenten in het PDF formaat konden worden gemaakt. Ik heb het prettige vermoeden dat we op dat vlak voorlopig nog wel een tijdje voorop zullen lopen. Ik wil niet zo ver gaan te stellen dat met TEX alles kan, maar de vaak onderschatte `\special` primitieve opent vooralsnog vele deuren. Voorlopig kan ik nog alles wat ik wil met TEX, hoewel de komst van e-TEX het leven wel wat gemakkelijker zou maken.[7]

10 Tot slot

Tot slot nog een woord ter relativering. Het is mogelijk om in TEX zeer geavanceerde en mooie documenten te maken, niet in de laatste plaats dankzij de beschikbare macropakketten. Daarbij kan een hoge mate van structuur worden gerealiseerd. Dit maakt enerzijds bemoeienis van de auteur met de opmaak overbodig en biedt anderzijds de vormgever eindeloos veel variaties in opmaak. Ik zie me vrijwel dagelijks in beide rollen geplaatst. Bovendien heb ik me de afgelopen jaren in de gelukkige omstandigheid bevonden een systeem te kunnen ontwikkelen, dat aan een groot aantal van de hierboven impliciet vermelde verwachtingen voldoet.

Het is gelukkig nog steeds mogelijk een maximum aan resultaat te bereiken met een minimum aan software. Kijkend naar de TEX-wereld doet me wel eens vermoeden dat hier het geheel niet altijd meer is dan de som der delen. Soms verlang ik dan ook wel eens terug naar de tijd dat wij gesoleerd van de rest van de TEX-wereld bezig waren. Aan de andere kant zijn juist de collectieve gedrevenheid en het in opofferingen resulterende enthousiasme van al die gebruikers een voortdurende bron van inspiratie. Ik krijg al hoofdpijn van de gedachte een ander systeem te moeten gebruiken dan TEX.

11 En de $\mathcal{NTG}$ dan?

De $\mathcal{NTG}$ is aan zichzelf verplicht af en toe haar eigen bestaansrecht te heroverwegen. Dit recht hangt nauw samen met het gebruik van TEX in Nederland, hoewel de landsgrenzen schijnen te vervagen. Het bestuur is dan ook van plan haar beleid voor de komende jaren te verwoorden in een strategisch plan. Ik heb hierboven getracht een aanzet te geven voor een brede discussie over het wel en wee en vooral de toekomst van het gebruik van TEX.

De bovenstaande uitspraken komen volledig voor mijn rekening. U kunt er het bestuur niet op aanspreken. Wel kunt u aan het bestuur kenbaar maken wat uw mening is. Grijp de pen of het toetsenbord en vertel ons wat u als $\mathcal{NTG}$-lid denkt. Vertel ons wat u met TEX doet, wat u van de TEX-wereld verwacht, hoe u de toekomst ziet en vooral wat de rol van de $\mathcal{NTG}$ moet zijn.

Waarom bent u bijvoorbeeld TEX gaan gebruiken. Was het omdat u wiskundig zetwerk moest opleveren of omdat het gratis was, was u op zoek naar efficiency, flexibiliteit en structuur of zocht u kwaliteit? Was het uit liefde of kwam het voort uit noodzaak?

Hoe ziet u het zelf? Is het voor u eigenlijk wel zinvol om door te gaan met het gebruiken van TEX of het daarop

[6] Hoewel die vlieger niet altijd opgaat voor TEX. Ik ben inmiddels zo gewend aan symmetrisch verbatim, buffers, verplaatsbare blokken, abstractie en andere structurerende gemakken, dat het gros van wat ik op papier zet de MAPS of TUGBOAT niet kan halen omdat conversie naar LATEX niet mogelijk is.

[7] En als ik klaar ben met het documenteren van ConTEXt, een klusje van enige duizenden bladzijden, hoop ik nog energie over te hebben om Ω te verkennen.

gebaseerde macropakket? Bent u niet net zo'n verstokte TEX-gebruiker als een ander dat is van zijn systeem? Zit u te wachten op macro's, technische ondersteuning of wilt u gewoon weten hoe u het best iets kunt doen? Is het zinvol een scheiding te maken tussen techniek en gebruik?

De activiteiten van de $\mathcal{NTG}$ kunnen worden gestructureerd in werkgroepen. Op dit moment zijn er twee actief. Een houdt zich bezig met educatie in brede zin en de andere buigt zich over spellingscontrole en afbreekpatronen. De eerste groep bestaat al lang, de tweede is nog maar net een echte groep. Heeft het zin om groepen in te richten rond specifieke pakketten, zoals TEX en $\mathcal{AMS}$-TEX, en formaten, zoals DVI, POSTSCRIPT en PDF, of moeten we vertrekken vanuit specifieke domeinen, zoals wiskunde en talen? Zijn twee bijeenkomsten per jaar genoeg of moeten er misschien meer gerichte bijeenkomsten komen?

Als u zich volledig kunt vinden in de huidige opzet en ondersteuning van de $\mathcal{NTG}$, dan is ook dat belangrijk om te horen. De $\mathcal{NTG}$ kan dan proberen nog beter te doen wat ze al doet.

Mijn laatste vraag is wellicht de meest essentile. Willen we eigenlijk wel dat TEX door iedereen wordt gebruikt? Is zo'n tool in zijn huidige vorm in veilige handen bij de gemiddelde gebruiker? Met TEX kunnen heel mooie dingen worden gemaakt, maar ook heel lelijke! Heeft exclusiviteit ook niet zo zijn voordelen? TEX is een zetsysteem en niet iedereen is typograaf. Rond de driehonderd leden zijn nog te overzien. We kennen elkaar en passen nog in n zaal. Streven we naar duizenden leden? En als ons dat ooit mocht overkomen, hoe richten we dan de vereniging in? Trouwens: hoeveel mensen gebruiken eigenlijk TEX? En: representeren zij die het meest actief zijn in vereniging wel de rest?

Do journals honor LaTeX submissions?

Gabriel Valiente Feruglio

Technical University of Catalonia
Departament of Software
E-08028 Barcelona, Catalonia, Spain
valiente@lsi.upc.es

Abstract

The survival of LaTeX in the academic world will depend on TeX and LaTeX evolving towards changing publishing practices, but also on publishers actually accepting LaTeX submissions. Much has been said about the former, with a recent feature number of *TUGboat* addressing the subject of electronic publishing. When it comes to the latter issue, however, it is often taken for granted that scientific journals honor LaTeX submissions.
An extensive research over the Internet reveals that, on the contrary, at the turn of the century many journals still regret to accept LaTeX submissions, sometimes preferring RTF or even bare ASCII sources to TeX or LaTeX.
This note discusses some of the issues behind this situation and compiles all journals known to the author that accept electronic submission of LaTeX articles in source form, thereby complementing the TeX counterpart [1].

1 Introduction

This note addresses the survival of LaTeX in the academic world, and it does it from the perspective of electronic publishing of LaTeX articles in scientific journals. Such a perspective is necessarily limited, since survival of LaTeX in the academic world will undoubtely depend on a multitude of factors, often intertwined, but it is quite interesting in itself since it will provide further motivation for PhD students, young scientists, and teaching assistants to adopt LaTeX as an integral solution for their typesetting needs along their academic lives, from writing a PhD thesis to typesetting class notes, research articles, and textbooks.

In fact, the original motivation for writing down this note was to attract potential LaTeX users among PhD students by showing them still another benefit of adopting LaTeX for their typesetting tasks, namely that scientific journals accept and encourage electronic submission of LaTeX sources. Such was also the motivation behind the chapter on electronic publishing in the author's recent LaTeX textbook [13].

An extensive research over the Internet was then conducted in order to find all journals that accept electronic submission of LaTeX articles in source form. Despite many journals not even mentioning the possibility for TeX or LaTeX submissions, the research shows that LaTeX use has spread well beyond the traditional subject areas of computer science, mathematics and physics.

Section 2 gives an overview of the whole process of LaTeX article submission, processing, and publishing. The results of the research over the Internet are summarized in Section 3 and they are discussed in Section 4. As a direct consequence of that discussion, the creation of a Technical Working Group to support and coordinate publisher's efforts is proposed in Section 5. The data resulting from the research over the Internet is presented in Appendix A.

2 Dynamics of LaTeX submissions

Submission of articles marked up with LaTeX may have different pros and cons for the people involved, from author and academic editor to reviewer and publisher. The whole process of submitting, processing and publishing a LaTeX journal submission is briefly reviewed in the following in order to put some of the issues involved in the right perspective:

1. The author writes a LaTeX article.
2. The author submits the article to one of the journal's academic editors.
3. The academic editor selects one or more reviewers and sends them the article.
4. The reviewers judge the article and advise the academic editor on acceptance.
5. The academic editor decides to accept the article, with or without changes, or to reject it.
6. On acceptance, the —probably revised— article is sent over to the publisher.
7. The publisher processes the article.
8. Although the author can obtain galley proofs (laser printer output), in some cases the publisher sends a page proof (phototypesetter output) to the author.
9. The publisher —usually a technical editor or a copy editor— applies final corrections to the article.

10. The article is included in a journal issue, either printed and/or electronic, and the issue is distributed.

Compared to traditional manuscript submission and processing, submission of LaTeX sources offers many advantages:

Faster delivery LaTeX sources can be sent by electronic mail or by `ftp`, a delivery method that is much faster than regular mail or even courier mail and much cheaper than the latter. This is an interesting issue, since an article is sent several times, at least three: author to academic editor, academic editor to each of the reviewers, and academic editor to publisher. It must be noticed, however, that editor and reviewers can still communicate by any means they choose about the review, including —but not limited to— further LaTeX sources[1], irrespective of whether the submission was a LaTeX source.

Reduced proof-reading Since there is no need of re-keying the submitted article from a paper copy, there is no real need for the publisher to send galley proofs to the author. No typing errors are (supposed to be) introduced in the article[2].

Shorter publication time Bypassing the typesetter and reducing or even eliminating proof-reading, production of page proofs is much faster and the overall cost of publication is reduced.

Reliability Whenever the publisher makes a LaTeX macro package available, the author can compile the article and obtain a preprint which is almost identical to the published article, perhaps differing only in page numbering and journal identification. Layout problems can be fixed by the author even before first submitting the article, contributing then to a further reduction in publication time and cost. The dark side of this issue is a burden on the author, who gets distracted from the article's content and becomes more of a copy editor.

Availability The author has an almost final version of the submitted article, which can be further distributed —usually in the form of a DVI or PostScript file— by electronic mail, `ftp`, the World-Wide Web (WWW), or a preprint archive [11]. This is indeed a highly controversial issue, since it affects the interests of the publisher, but as long as authors do not transfer copyright to publishers they are entitled to, say, put their articles in their WWW home pages. Some kind of balance will surely have to be found between author's interest in having their work as broadly disseminated as possible and publisher's economic interest which makes such a dissemination possible[3].

There are, however, some disadvantages to the submission and processing of LaTeX sources:

Processing burden Processing the LaTeX submission by academic editor and reviewers can be much of a burden on them. They need to assure that they get the complete submission, which often consists of several LaTeX source files and a set of EPS illustrations. The submission may fail to compile due to missing parts, required LaTeX macro packages not available at their installation, errors in included EPS figures, etc. It should not be overseen that most academic editors and almost all reviewers are not paid for their services.

Investment in learning Publishing staff and typesetters need to invest in learning TeX —which shows a steep learning curve— and in setting up and maintaining a whole TeX system, including high-resolution output devices and their drivers, integration of text and images, etc.

Some of these issues may explain why many journals accept and process LaTeX submissions but in most cases the academic editors prefer paper submissions; see the discussion in Section 4 below.

3 Journals

Finding out those journals that accept electronic submission of articles marked up with LaTeX would have not been possible if publishers did not offer journal information on the Internet. As a matter of fact, most publishers already maintain home pages for their journals on the World-Wide Web, and in many cases these pages offer extensive information for authors.

The following list gives the number of journals found within each scientific field that accept LaTeX submissions, according to the *Science and Engineering Field Classification* made by the National Science Foundation. The classification scheme is available at `http://www.qrc.com/nsf/srs/rdexp/`.

- Computer Sciences 97
- Mathematical Sciences 89
- Engineering 77

[1] In the case of the *Rewriting Techniques and Applications* conferences, for instance, review reports are standard LaTeX document templates which the conference organizers send to the reviewers, who fill them in and send back to the organizers, who then send over to the authors, and the whole process takes place over electronic mail.

[2] During the review of the book "On Being a Machine, Vol. 1: Formal Aspects of Artificial Intelligence," by A. Narayanan (Ellis Horwood, 1988) I had found over 300 typographic mistakes which the author attributed to the publisher's re-keying of the submission. A. Narayanan moved then to LaTeX and provided Ellis Horwood with camera-ready copies for the second volume, "On Being a Machine, Vol. 2: Philosophy of Artificial Intelligence" (Ellis Horwood, 1990). The review appeared in *Artificial Intelligence* 12(4):96–97, 1991.

[3] A first step in this direction has been taken recently by Elsevier Science for the *Electronic Notes in Theoretical Computer Science* series of Conference Proceedings, whereby authors are forbidden to make their contributions available by anonymous `ftp` or ever the WWW but are allowed instead to include links from their WWW pages to Elsevier Science's own WWW pages, where full access to articles is only granted to people accessing from an institution which holds a subscription to the *Theoretical Computer Science* journal.

- Physical Sciences 61
- Life Sciences 51
- Environmental Sciences 18
- Social Sciences 18
- Other Sciences (Multidisciplinary) 7

  Total 418

As can be seen from the previous list, adoption of LaTeX in scientific publishing has spread well beyond the traditional subject areas of computer science, mathematics and physics. Notice, however, that for each journal accepting submissions of articles marked up with LaTeX there may be up to ten journals in the same field which do not accept LaTeX submissions.

4 Discussion

Some of the issues behind the situation described in section 3 are depicted in the following in the form of short *provocative statements*, which are not meant to be definitive assertions but to rather spark further debate within the TeX community about the future of LaTeX in the academic world.

Publishers regret to accept LaTeX submissions because it doesn't pay off

Let alone publishers who have never heard about LaTeX, even for those who care about LaTeX keeping up with LaTeX developments may represent too big an overhead. Take for instance Springer Verlag, who has even replaced its well-known `llncs` macro package by the LaTeX 2.09 formats (NFSS version 1) *CLMono01* and *CLMult01*.

As a matter of fact, the proof is that almost two years after the first release of LaTeX2$_\varepsilon$, relatively few scientific publishers have updated their LaTeX macro packages to LaTeX2$_\varepsilon$.

Moreover, many publishers argue that setting up a TeX system, keeping it up-to-date, and polishing LaTeX submissions to match their house styles is usually more expensive and time-consuming than re-keying the submitted articles from author-supplied hard copies.

Publishers do not get articles marked up with LaTeX for publication

One of the reasons why most publishers in the fields of environmental, life, and social sciences do not honor LaTeX submissions is that they rarely get articles marked up with LaTeX for publication. As a matter of fact, authors seem to be the driving force behind the adoption of LaTeX by scientific publishers.

Publishers force authors to submit *standard* LaTeX articles

Publishers complain that it is almost impossible to have authors submit articles marked up with *standard* LaTeX, that is, without author-defined macros, while authors complain that publishers limit their creativity by forcing them to comply with some LaTeX macro package [8]. Maybe both sides are right in their complaints, but the truth is that publishers have a good deal of work at polishing LaTeX submissions and resolving macro name clashes, while it is both unreasonable and contrary to LaTeX's philosophy to forbid authors defining new macros in their articles.

A solution to both sides of the problem can be foreseen in the form of either an extension to the LaTeX kernel, a macro package or some kind of utility program, which would expand all author-defined macros and output a *standard* LaTeX article source. The question is, what exactly is a *standard* LaTeX article source?

Journals honor LaTeX submissions but academic editors do not

Although many publishers have all the hardware, software and know-how needed to process LaTeX submissions, however, academic editors for each of the journals they publish always have the last word.

Take, for instance, some of the major scientific publishers which are moving into electronic publication [12]. Elsevier Science accepts, in principle, LaTeX submissions for all of its 1100 journals but academic editors for only 7 of them are willing to accept LaTeX submissions.

A similar pattern is repeated for other publishers. Academic editors at Springer Verlag only accept LaTeX submissions for 8 of its 350 journals, at John Wiley & Sons only 9 out of 326 journals do, at Blackwell Science only one out of 200 journals does, and at Academic Press only two out of 175 journals accept LaTeX submissions.

The question is, why do most academic editors desencourage submission of articles marked up with LaTeX, even though publishers provide them with running TeX systems and house styles already encoded in LaTeX macro packages?

Journals may no longer honor LaTeX submissions as they move electronic

Electronic journals, as well as preprint databases [11], accept any ASCII submission but in most cases prefer TeX or LaTeX, at least in the fields of engineering and computer, mathematical, and physical sciences. When it comes to environmental, life, and social sciences, however, it is much more common to find journals which only accept either RTF or HTML submissions.

LaTeX to HTML conversion may be seen as a practical solution. `LaTeX2HTML` [4] even allows the inclusion of hypertext links in articles. In practice, however, it may sacrifice typographical quality, since all mathematical formulas, figures and tables are converted to GIF (Graphics Interchange Format) images or PostScript pictures, which in most cases have a low resolution and cannot be zoomed in and out without distorting the image.

Besides, `LaTeX2HTML` fragments a well-structured LaTeX document into too many little files. Although the degree of splitting can be controlled by a parameter, it is set to a high value by default and, in practice, this turns

reading the document with an HTML browser into a kind of...

As HTML develops into HTML3, with some degree of support for mathematics and tables, it is possible that HTML takes over as the preferred format for submission to electronic journals in the fields of engineering and computer, mathematical, and physical sciences as well.

Conversion of TeX and LaTeX into SGML [9, 2] may help to avoid HTML ever displacing LaTeX as one of the preferred formats for submitting articles to scientific journals, since the scientific publishing industry seems to be moving definitely towards SGML.

5 Conclusion

An author may have to deal with many publishers, and therefore may need to comply with different TeX macro packages and instructions to authors. Adoption of LaTeX by an author may prove to be, in that sense, a rewarding decision as long as publishers encode their house styles in LaTeX macro packages. This would let authors concentrate on scientific content while keeping LaTeX training needs down to a point somewhere between [7] and [5].

An ideal situation would be for the author to write a standard `article`-class LaTeX document and to later add a

```
\usepackage{publisher}
```

mark, or even better a

```
\usepackage[journal]{publisher}
```

mark, right before submitting it to the publisher.

In practice, however, complying with the author instructions for a particular journal may involve various changes to the original LaTeX source, ranging from low-level font selection to high-level macros for theorem-like environments, inclusion of encapsulated PostScript figures, and author affiliation.

Such a high degree of transparency of publisher styles with respect to the standard LaTeX `article`-class can only be reached by a serious standardization effort. Maybe the time has come for the TeX Users Group to set up a new Technical Working Group (TWG), with the goal of coordinating publishers' efforts at encoding their journal styles in LaTeX macro packages. Such a TWG should also liason with the LaTeX3 Project Team in order to enhance the standard LaTeX `article.cls` document class and perhaps also `book.cls` and `report.cls`, by including more structural information in the front matter which would offer a standard interface to authors and could also be easily adapted to the particular needs of different publishers. As a matter of fact, some publisher packages that show the need for such an enhancement have been available for several years, among which Springer [10], Elsevier Science [3], DANTE [6], and many others.

In any case, the author sincerely hopes not to be charged with the whole task just because of having had such a bright idea.

Acknowledgement

I am very grateful to Barbara Beeton, Sebastian Rahtz and Christina Thiele for early comments on the very idea of this paper, and to the anonymous referees, whose suggestions have led to a substantial improvement of the article.

References

[1] Nelson Beebe. *Bibliography of Journals accepting Manuscripts written using TeX*. Electronic document available at `http://www.tex.ac.uk/tex-archive/info/biblio/texjourn.ltx`, 1994.

[2] Anne Brüggemann-Klein. Wissenschaftliches publizieren im umbruch. *Informatik—Forschung und Entwicklung*, 10:171–179, 1995.

[3] Elsevier. *Preparing Articles with LaTeX: Instructions to Authors for preparing Compuscripts.* Electronic document available at `http://www.tex.ac.uk/tex-archive/macros/latex/contrib/supported/elsevier/`, 1995.

[4] Michel Goossens and Janne Saarela. TeX to HTML and back. *TUGboat*, 16(2):174–214, 1995.

[5] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, 2nd edition, 1994.

[6] Gerd Neugebauer. Eine klasse für die TeXnische komödie. *Die TeXnische Komödie*, 4/95:6–15, 1996.

[7] Tobias Oetiker, Hubert Partl, Irene Hyna, and Elisabeth Schlegl. *The not so short Introduction to LaTeX2ε*. Electronic document available at `http://www.tex.ac.uk/tex-archive/info/lshort/`, 1995.

[8] Nico Poppelier. Two sides of the fence. *TUGboat*, 12(3):353–358, 1991.

[9] Sebastian Rahtz. Another look at LaTeX to SGML conversion. *TUGboat*, 16(3):315–324, 1995.

[10] Springer-Verlag. *Instructions for Authors using LaTeX and the Springer Macro Package CLMono01 or CLMult01*. Electronic document available at `ftp://trick.ntp.springer.de/pub/tex/latex/clmomu01/`, 1995.

[11] Gary Taubes. Electronic preprints point the way to author empowerment. *Science*, 271(5250):767, February 1996.

[12] Gary Taubes. Science journals go wired. *Science*, 271(5250):764, February 1996.

[13] Gabriel Valiente. *Composició de textos científics amb LaTeX*. Edicions UPC, Barcelona, 1996.

A Journals accepting manuscripts marked up with LaTeX

This appendix lists journals for which at least one of the editors accepts electronic submissions written using LaTeX, grouped by publisher. An HTML version of this list is available on the Internet at the address `http://www-lsi.upc.es/~valiente/journals.html` that links about 40 publishers and

more than 400 journals to their home pages on the World-Wide Web. Any help to bring it more complete and to keep it up-to-date is warmly welcome.

Academia Scientiarum Fennica
- Annales Academiæ Scientiarum Fennicæ

Academic Press
- Analytical Biochemistry
- J. of Approximation Theory

American Astronomical Society
- Astrophysical J.
- Astrophysical J. Supplement
- Astrophysical J. Letters
- Astronomical J.

American Institute of Physics
- The J. of the Acoustical Society of America

American Mathematical Society
- Bulletin of the $\mathcal{AMS}$
- Electronic Research Announcements of the $\mathcal{AMS}$
- J. of the $\mathcal{AMS}$
- Mathematics of Computation
- Notices of the $\mathcal{AMS}$
- Proc. of the $\mathcal{AMS}$
- Trans. of the $\mathcal{AMS}$

American Physical Society
- Physical Review A
- Physical Review B
- Physical Review C
- Physical Review D
- Physical Review E
- Physical Review Letters
- Reviews of Modern Physics

Association for Computing Machinery
- ACM Trans. on Mathematical Software
- Comm. of the ACM
- J. of the ACM
- IEEE/ACM Trans. on Networking
- J. of Experimental Algorithmics
- Trans. on Computer Systems
- Trans. on Computer-Human Interaction
- Trans. on Design Automation of Electronic Systems
- Trans. on Graphics
- Trans. on Information Systems
- Trans. on Mathematical Software
- Trans. on Modeling and Computer Simulation
- Trans. on Prog. Languages and Systems

Birkhäuser Verlag
- Aequationes Mathematicae
- Algebra Universalis
- Aquatic Sciences
- Archiv der Mathematik
- Botanica Helvetica
- Chemoecology
- Circuits, Systems, and Signal Processing
- Commentarii Mathematici Helvetici
- Computational and Applied Mathematics
- Computational Complexity
- Eclogae Geologicae Helvetiae
- Elemente der Mathematik
- EXPERIENTIA
- Fresenius Environmental Bulletin
- Geometric and Functional Analysis
- Helvetica Physica Acta
- Inflammation Research
- Insectes Sociaux
- Integral Equations and Operator Theory
- J. of Evolutionary Biology
- J. of Geometry
- J. of Mathematical Systems, Estimation, and Control
- MapleTech
- Medical Microbiology Letters
- Medicine
- Nonlinear Differential Equations and Applications
- NTM
- Pure and Applied Geophysics
- Resultate der Mathematik
- Selecta Mathematica, New Series
- Sozial- und Präventivmedizin
- Zeitschrift für angewandte Mathematik und Physik

Blackwell Publishers
- Computer Graphics Forum

Cameron University, Oklahoma
- Southwest J. of Pure and Applied Mathematics

Chapman & Hall
- Optical and Quantum Electronics

Computer Society of South Africa
- The South African Computer J.

Deutsche Mathematiker-Vereinigung
- Documenta Mathematica

DANTE
- Die TEXnische Komödie

Elsevier Science
- Artificial Intelligence
- Discrete Applied Mathematics
- Discrete Mathematics
- Electronic Notes in Theoretical Computer Science
- Linear Algebra and its Applications
- New Astronomy
- Theoretical Computer Science

Heldermann Verlag Berlin
- Beiträge zur Algebra und Geometrie
- J. of Lie Theory

Institute of Electrical and Electronics Engineers

- Computer
- IEEE Annals of the History of Computing
- IEEE Computational Science & Engineering
- IEEE Computer Graphics and Applications
- IEEE Design & Test of Computers
- IEEE Electron Device Letters
- IEEE Expert
- IEEE J. on Selected Areas in Communications
- IEEE J. on Selected Topics in Quantum Electronics
- IEEE J. of Microelectromechanical Systems
- IEEE J. of Quantum Electronics
- IEEE J. of Solid-State Circuits
- IEEE Micro
- IEEE Microwave and Guided Wave Letters
- IEEE MultiMedia
- IEEE Parallel & Distributed Technology
- IEEE Photonics Technology Letters
- IEEE Signal Processing Letters
- IEEE Software
- IEEE Trans. on Antennas and Propagation
- IEEE Trans. on Applied Superconductivity
- IEEE Trans. on Automatic Control
- IEEE Trans. on Biomedical Engineering
- IEEE Trans. on Circuits and Systems for Video Technology
- IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications
- IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing
- IEEE Trans. on Communications
- IEEE Trans. on Components, Packaging, and Manufacturing Technology Part A
- IEEE Trans. on Components, Packaging, and Manufacturing Technology Part B
- IEEE Trans. on Components, Packaging, and Manufacturing Technology Part C
- IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems
- IEEE Trans. on Computers
- IEEE Trans. on Control Systems Technology
- IEEE Trans. on Education
- IEEE Trans. on Electromagnetic Compatability
- IEEE Trans. on Electron Devices
- IEEE Trans. on Engineering Management
- IEEE Trans. on Fuzzy Systems
- IEEE Trans. on Geoscience and Remote Sensing
- IEEE Trans. on Image Processing
- IEEE Trans. on Industrial Electronics
- IEEE Trans. on Industry Applications
- IEEE Trans. on Information Theory
- IEEE Trans. on Instrumentation and Measurement
- IEEE Trans. on Knowledge & Data Engineering
- IEEE Trans. on Magnetics
- IEEE Trans. on Medical Imaging
- IEEE Trans. on Mechatronics
- IEEE Trans. on Microwave Theory and Techniques
- IEEE Trans. on Neural Networks
- IEEE Trans. on Nuclear Science
- IEEE Trans. on Oceanic Engineering
- IEEE Trans. on Parallel & Distributed Systems
- IEEE Trans. on Pattern Analysis & Machine Intelligence
- IEEE Trans. on Plasma Science
- IEEE Trans. on Power Electronics
- IEEE Trans. on Professional Communication
- IEEE Trans. on Rehabilitation Engineering
- IEEE Trans. on Robotics and Automation
- IEEE Trans. on Semiconductor Manufacturing
- IEEE Trans. on Signal Processing
- IEEE Trans. on Software Engineering
- IEEE Trans. on Speech and Audio Processing
- IEEE Trans. on Systems, Man, and Cybernetics Part A: Systems and Humans
- IEEE Trans. on Systems, Man, and Cybernetics Part B: Cybernetics
- IEEE Trans. on Ultrasonics, Ferroelectrics, and Frequency Control
- IEEE Trans. on Very Large Scale Integration (VLSI) Systems
- IEEE Trans. on Visualization & Computer Graphics
- IEEE Trans. on VLSI Systems
- IEEE/ACM Trans. on Networking
- IEEE/OSA J. of Lightwave Technology
- Proc. of the IEEE

Institute of Physics Publishing

- Bioimaging
- Classical and Quantum Gravity
- Distributed Systems Engineering
- European J. of Physics
- High Performance Polymers
- Inverse Problems
- J. of Micromechanics and Microengineering
- J. of Physics A: Mathematical and General
- J. of Physics B: Atomic, Molecular and Optical Physics
- J. of Physics: Condensed Matter
- J. of Physics D: Applied Physics
- J. of Physics G: Nuclear and Particle Physics
- J. of Radiological Protection
- Measurement Science and Technology
- Modelling and Simulation in Materials Science and Engineering
- Nanotechnology
- Network: Computation in Neural Systems
- Nonlinearity
- Physics Education
- Physics in Medicine and Biology
- Physiological Measurement
- Plasma Physics and Controlled Fusion
- Plasma Sources Science and Technology
- Public Understanding of Science
- Pure and Applied Optics
- Quantum and Semiclassical Optics
- Reports on Progress in Physics
- Semiconductor Science and Technology
- Smart Materials and Structures
- Superconductor Science and Technology
- Waves in Random Media

IOS Press

- AI Communications
- Asymptotic Analysis
- BioFactors
- Bio-Medical Materials and Engineering
- Chinese Science Bulletin (Kexue Tongbao)
- Education for Information
- Environmental Policy and Law
- Fundamenta Informaticæ
- Human Systems Management

- Information and Systems Engineering
- Information Infrastructure and Policy
- Information Services and Use
- Information Technology for Development
- Int. J. of Applied Electromagnetics and Mechanics
- Int. J. of Risk and Safety in Medicine
- J. of Computer Security
- J. of Economic and Social Measurement
- J. of Environmental Sciences
- J. of High Speed Networks
- Pharmacotherapy
- Reviews in Toxicology
- Space Communications
- Spectroscopy: An Int. J.
- Statistical J. of the United Nations Economic Commission for Europe
- Technology and Health Care

Kent State University

- Electronic Trans. on Numerical Analysis

Kluwer Academic Publishers

- Acta Applicandae Mathematicae
- Adsorption
- Analog Integrated Circuits and Signal Processing
- Applied Cardiopulmonary Pathophysiology
- Applied Categorical Structures
- Applied Composite Materials
- Applied Intelligence
- Applied Scientific Research
- Aquatic Geochemistry
- Archives of Suicide Research
- Astrophysics and Space Science
- Automated Software Engineering
- Autonomous Robots
- Biodegradation
- Biogeochemistry
- Bioseparation
- Biotherapy
- Boundary-Layer Meteorology
- Celestial Mechanics and Dynamical Astronomy
- Climatic Change
- Compositio Mathematica
- Computational Economics
- Computational Optimization and Applications
- Computers and the Humanities
- Crime, Law and Social Change
- Cytotechnology
- Design Automation for Embedded Systems
- Designs, Codes and Cryptography
- Discrete Event Dynamic Systems
- Distributed and Parallel Databases
- Documenta Ophthalmologica
- Dynamics and Control
- Earth, Moon and Planets
- Economics of Planning
- Educational Studies in Mathematics
- Empirica
- Entomologia Experimentalis et Applicata
- Environmental Monitoring and Assessment
- Euphytica
- European J. of Health Law
- European J. of Population
- Experimental Astronomy
- Financial Engineering and the Japanese Markets
- Formal Methods in System Design
- Gazette
- Genetic Resources and Crop Evolution
- Genetica
- Geology and Mining (Geologie en Mijnbouw)
- Geometriae Dedicata
- Geriatric Nephrology and Urology
- Hydrobiologia
- Instructional Science
- Interface Science
- Int. J. of Clinical Monitoring and Computing
- Int. J. of Computer Vision
- Int. J. of Fracture
- Int. J. of General and Molecular Microbiology
- Int. J. of Salt Lake Research
- Int. J. of Value-Based Management
- Int. J. on Group Rights
- Int. Ophthalmology
- J. for General Philosophy of Science
- J. of Algebraic Combinatorics
- J. of Applied Phycology
- J. of Aquatic Ecosystem Health
- J. of Atmospheric Chemistry
- J. of Automated Reasoning
- J. of Biological Physics
- J. of Elasticity
- J. of Electronic Testing
- J. of Engineering Mathematics
- J. of Global Optimization
- J. of Inclusion Phenomena and Molecular Recognition in Chemistry
- J. of Intelligent Information Systems
- J. of Logic, Language and Information
- J. of Mathematical Imaging and Vision
- J. of Paleolimnology
- J. of Sol-Gel Science and Technology
- J. of Systems Integration
- K-Theory
- Letters in Mathematical Physics
- Lifetime Data Analysis
- LISP and Symbolic Computation
- Machine Learning
- Machine Translation
- Man and World
- Meccanica
- Medical Progress Through Technology
- Molecular Biology Reports
- Multidimensional Systems and Signal Processing
- Multimedia Tools and Applications
- Mycopathologia
- Natural Hazards
- New Forests
- Nonlinear Dynamics
- Nutrient Cycling in Agroecosystems
- Origins of Life and Evolution of the Biosphere
- Philosophical Studies
- Photosynthesis Research
- Plant and Soil
- Plant Cell, Tissue and Organ Culture
- Plant Growth Regulation
- Potential Analysis
- Real-Time Systems

- Review of Industrial Organization
- Set-Valued Analysis
- Social Indicators Research
- Solar Physics
- Studies in East European Thought
- Surveys in Geophysics
- Systematic Parasitology
- The EDI Law Review
- The J. of Supercomputing
- The J. of VLSI Signal Processing
- The Int. J. of Cardiac Imaging
- Transport in Porous Media
- User Modeling and User-Adapted Interaction
- Vegetatio
- Water Resources Management
- Water, Air and Soil Pollution

Masaryk University, Czech Republic
- Archivum Mathematikum

Morgan Kaufmann
- J. of Artificial Intelligence Research

Optical Society of America
- Applied Optics
- J. of the Optical Society of America A
- J. of the Optical Society of America B
- Optics Letters
- J. of Lightwave Technology
- Chinese J. of Lasers B
- J. of Optical Technology
- Optics & Spectroscopy

Oxford University Press
- The Computer J.

Royal Astronomical Society
- Monthly Notices of the Royal Astronomical Society

Sociedad Colombiana de Matemáticas
- Revista Colombiana de Matemáticas

Societat Catalana de Matemàtiques
- SCM/Notícies

Société de Mathématiques Appliquées et Industrielles
- ESAIM: Control, Optimisation and Calculus of Variations
- ESAIM: Probability and Statistics
- ESAIM: Proc.

Society for Industrial and Applied Mathematics
- SIAM J. on Applied Mathematics
- SIAM J. on Computing
- SIAM J. on Control and Optimization
- SIAM J. on Discrete Mathematics
- SIAM J. on Mathematical Analysis
- SIAM J. on Matrix Analysis and Applications
- SIAM J. on Numerical Analysis
- SIAM J. on Optimization
- SIAM J. on Scientific Computing
- SIAM Review

Springer Verlag
- Constructive Approximation
- Few-Body Systems Electronic
- Informatik—Forschung und Entwicklung
- J. of Nonlinear Science
- J. of Universal Computer Science
- J. of Very Large Databases
- Numerische Mathematik Electronic Edition
- Semigroup Forum

TeX Users Group
- TeX and TUG News
- TUGboat

The International Linear Algebra Society
- Electronic J. of Linear Algebra

Universidad Nacional Autónoma de México
- Revista Electrónica del Departamento de Matemáticas

University at Albany, State University of New York
- New York J. of Mathematics

Univerzita Komenskeho, Bratislava
- Acta Mathematica Universitatis Comenianæ

The Johns Hopkins University Press
- American J. of Mathematics

The MIT Press
- Artificial Life
- Computational Linguistics
- Evolutionary Computation
- J. of Functional and Logic Prog.
- Neural Computation
- The Chicago J. of Theoretical Computer Science
- The Int. J. of Robotics Research

John Wiley & Sons
- Comm. in Numerical Methods in Engineering
- Electronic Publishing: Origination, Dissemination and Design
- J. of Combinatorial Designs
- J. of Graph Theory
- Int. J. for Numerical Methods in Engineering
- Naval Research Logistics
- Numerical Methods for Partial Differential Equations
- Random Strucrures and Algorithms
- Theory and Practice of Object Systems

World Scientific

- Int. J. of Cooperative Information Systems
- Int. J. of Foundations of Computer Science
- Int. J. of High Speed Computing
- Int. J. of High Speed Electronics and Systems
- Int. J. of Information Technology
- Int. J. of Modern Physics A: High Energy Physics
- Int. J. of Modern Physics B: Condensed Matter Physics
- Int. J. of Modern Physics C: Computational Physics
- Int. J. of Modern Physics D: Astrophysics
- Int. J. of Modern Physics E: Nuclear Physics
- Int. J. of Reliability, Quality and Safety Engineering
- Int. J. of Shape Modeling
- Int. J. of Software Engineering and Knowledge Engineering
- Int. J. of Uncertainty, Fuzziness and Knowledge-Based Systems
- Int. J. on Artificial Intelligence Tools
- J. of Circuits, Systems and Computers
- J. of Computational Acoustics
- J. of Knot Theory and its Ramifications
- Mathematical Models and Methods in Applied Sciences
- Modern Physics Letters A: High Energy Physics
- Modern Physics Letters B: Condensed Matter Physics
- Parallel Processing Letters

Other

- BIT
- Electronic J. of Combinatorics
- Electronic J. of Differential Equations
- Electronic J. of Probability
- Electronic Comm. in Probability
- Theory and Applications of Categories
- Reliable Computing

Introduction to "TEX Unbound: LATEX & TEX Strategies Fonts, Graphics, and More"

Alan Hoenig *

The reader contemplating this book has a right to know what the author's goals are—or are not—for this volume. The non-goals are easy—this book will *not* discuss the various typesetting commands supported by TEX and LATEX (except casually or in passing). It's much better to consult the canonical works by Knuth or Lamport, or any of the several excellent books this canon has inspired, for that kind of information.

Anyone who knows the slightest bit about typesetting with LATEX or TEX knows there is more to fine typesetting than the commands of the TEX language. It's important to know this material well, but there are other issues concerning document production that the canon barely touches upon. For instance:

- How can I make full use of the many commercial, digital fonts? And how may I use them to typeset technical texts in a fully professional manner? And if I use these fonts, can I also typeset mathematics in a visually compatible way?
- What about graphics—how may I prepare and include images and graphic material for my TEX document?
- Do the Internet, multimedia, and hypertext have any relevance to TEX (and vice versa)?
- Can TEX be made to fit into the suite of general office and educational software that is so ubiquitous, or is TEX *sui generis*?
- Low level query: what is TEX and why should I care? What is LATEX, and how does it differ from TEX? Where did TEX come from? How do I best bring myself up to speed as a TEX user?

It is my intention to provide discussions to these and similar queries in the pages that follow.

These are disparate issues, though, linked only by their absence in a standard 'TEXtbook'. Some of this material is elementary, while other bits are quite advanced. As a result, it's hard to fix a single label on this book as to level—it is neither elementary, nor intermediate, nor advanced, but all three at once.

The three parts of the book attempt dealing with these issues. In the first part, we present surveys of useful areas—what computer typesetting and TEX involve exactly; what Internet resources there are for the TEX-aware author; introductions to Metafont and METAPOST, TEX's graphic siblings; logical document structure (including SGML) and LATEX; and some tips and suggestions for using TEX alongside standard office and academic software (but a discussion of HyperTEXt also appears here).

The lengthy second part is a discussion of virtual fonts, but it begins with a discussion of font installation and selection for both plain TEX and LATEX, so authors will be able to use non-Computer Modern fonts in their TEX documents. After an extensive examination of the virtual font concept, several chapters present instructions for carrying out many virtual font projects:

- simple DC font creation;
- installing outline fonts for use by TEX and LATEX;
- creating real small caps fonts;
- oblique (slanted) and unslanted italic fonts;
- old style figures in fonts;
- better footnote numbers with expert fonts;
- introduction to foreign language typesetting;
- underlining and striking-out of extensive passages of text;
- bold fonts when no bold font exists;
- f-words (words that end in f);
- alternate fonts containing special characters and exotic ligatures;
- kern tracking and letterspacing;
- previewing output using that contains PostScript fonts;
- hints and suggestions for properly scaling fonts at different sizes; and
- creating and installing new math packages, so authors can properly typeset mathematics using MathTime, Lucida, Euler, or Mathematica math fonts, plus sans serif, typewriter, fraktur, calligraphic, and blackboard bold fonts.

The 'new math' section contains an extensive "rogues' gallery" showing how combining various roman faces with math fonts leads to different visual effects.

The final portion of the book addresses some fun questions—how do you create and place graphic images in a document? There are many excellent tools to accomplish this, but even with the limited discussion we're restricted

*This book will be published by Oxford University Press in early 1997. Contact the author at `ajhjj@cunyvm.cuny.edu` for further information.

to, there's lots to say. After some general discussion, we focus on four very special approaches:

- the LATEX picture environment (part of LATEX) and extensions thereto;
- Metafont and METAPOST; and
- two packages powered by TEX front-ends, PSTricks (a TEX front-end to the PostScript language), and MFPic (a TEX front-end to Metafont).

Two appendices present whirlwind introductions of TEX and LATEX, and one discusses the techniques used to produce this book (which was produced entirely by TEX or its siblings and friends).

What's the best way to use this book? The author hopes that everyone will find the time to sit back, relax, and read everything from cover to cover, but this is not practical for most readers. Please do, though, take time to riffle through this volume. Note that several chapters conclude with compendia or list of commands which just may prove useful. Note too that the entries in the bibliography have back references, so it's possible to find out where in the book a citation received discussion. Finally, ignore the index at your peril! Every effort has been made to make the index complete and consistent. Who would guess that discussion of underlining appears in a virtual font chapter—but the index points you to the proper page.

One principle silently informs much of the book's discussion. I call this "Hayes's Principle of Software."

> No matter how many palettes of buttons and how many menu options are offered, users of a program will always want to do something the author has not foreseen. Adding still more buttons and menus is not the answer.

This is why TEX (or a comparable descendent program) will endure in the face of huge advertising efforts by software giants. But there's another result of Hayes's Principle—authors of TEX documents tend to be on their own private cutting edge in their inadvertant approach to the unforeseen. One final purpose of this book is to enhance the insight of a TEX user, who, while now conscious of ever more things to do within TEX, will now know ever more ways to carry them out.

Virtual Fonts, Virtuous Fonts

Alan Hoenig

Abstract

Virtual fonts allow us to use all digital fonts with TEX, even non-TEX ones, and do much more for us. What are virtual fonts? Several projects grant us necessary experience with them.
This document comprises somewhat less than half of the similarly named chapter which will appear in the book **TEX Unbound: LATEX and TEXStrategies for Fonts, Graphics, and More**, by Alan Hoenig, to be published in early 1997 by Oxford University Press. This excerpt is simplified so that it may be printed using the standard suite of TEX fonts; the original depends heavily on PostScript fonts and the author's style file for its typesetting. Consequently, some displays could not be included. For any questions or comments, please contact the author at `ajhjj@cunyvm.cuny.edu`.

When talking about computers, we use the adjective "virtual" to describe a thing that behaves like something else. Virtual disks are really memory blocks which simulate hard disks, while virtual memory uses a disk to mimic a computer's memory. A virtual font looks to TEX like any other font, but it really is pieced together from other fonts or collections of typographic elements. It may be

- a composite of several different fonts somehow mixed together (in a special way, according to precise rules);
- a single font whose characters are (for very good reasons) scrambled in some new order;
- a collection of constructed characters, each built from several components (like accented letters are), which behaves like a font;
- individual horizontal or vertical rules each of which is treated as a character in a font;
- a collection of text, graphics, or PostScript files, each of which is treated as a single character within the virtual font;
- a conglomerate of all (or some) of the above.

This and the next few chapters explore virtual fonts, consider occasions that need them, and provide procedures for constructing them. It's messy constructing virtual fonts by hand, but a few freely available resources make it easy.

The box (not included here—sorry) lists some virtual font projects. For most people, the only application of virtual fonts may be to perform the proper installation of outline fonts (that is, PostScript fonts) for use by TEX. (We will use the term *installation* to describe the entire process of making fonts usable by TEX.) Many tasks difficult or impossible to accomplish with macros become trivial when implemented via virtual fonts.

The next few sections explain the concept of virtual font, together with the related concepts of font tables and encoding tables, in detail. Thereafter, we will provide discussion and procedures for implementing most of the applications in the list above.

1 The virtual font concept

Let's begin by journeying to a different planet, one on which a system like TEX has been developed, but on which all languages contain only two distinct characters, which we can call 'e' and 'f', together with the double-f ligature 'ff'. A close examination of a font on this hypothetical planet makes it easier to understand the kinds of problems arising in real, terrestrial fonts, and how virtual fonts can solve them.

A table listing the characters of any TEX font would contain only three characters.

0	1	2
e	**f**	**ff**

These three characters have been numbered using the usual computer science convention which starts with 0. These numeric labels serve to identify the position in the font of each character.

These numeric positions also play an important role for TEX, for `dvi` files contain typesetting commands based *not* on the glyph name ('A', 'B', 'comma', or whatever) but on each numeric label. For any 'e' in the input file, the `dvi` file contains the instruction to typeset character 0 in the current font. The lowercase 'e' had better be in that position! This correspondence between character and character number is built into the TEX program, and that's true for both the distant planet and for ours.

Difficulties arise when we try to use a commercial font instead of Computer Modern. We will suppose that the commercial font we want contains three characters, but they are 'e', 'f', and '&'. To get the ligature, we need to purchase a separate font, which contains the 'ff' plus two other characters. A further difficulty surfaces when we examine the layout of the fonts.

0	1	2
f	**e**	**&**

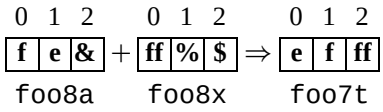
0	1	2
ff	**%**	**$**

The characters in the main font are in the wrong order, and this leads to disaster. To see why, let's select the commercial font, and now suppose we type `f`. TEX expects an 'f' to occupy position 1 of the font table, and so puts an instruction (in the `dvi` file) to typeset character 1. But character 1 in the non-TEX font is the glyph 'e', and that's what gets typeset—not the 'f' that we requested. Moreover, it does not appear that we can typeset the ff without an explicit call to the auxiliary font. Apparently, the input file will look different whether we typeset with the usual TEX fonts or with some other fonts, and this is unacceptable.

Virtual fonts have been created to deal with this (and other) exigencies. As far as an author is concerned, a virtual font is just another font. But it provides a mechanism whereby (behind the scenes), real fonts (*raw fonts*) can be combined so that the resulting *virtual font* conforms to the usual TEX conventions to eliminate any need for marking up the input file differently. In our example, a virtual font would

1. select the e and f from the main font, and re-order them in a TEX-acceptable way; and
2. include the ligature from the expert font in the last position in the table for the virtual font.

We call an auxiliary font containing ligatures and other special symbols an expert font. Furthermore, we'll follow the terrestrial convention of labelling raw fonts by appending '8a' or '8x' (expert) to them. So raw font `foo8a` and expert font `foo8x` come together in the virtual font `foo7t`. (Chapter 6 explains the conventions surrounding the notations 8a, 8x, and 7t.)



Font `foo7t` uses selected characters from the two raw fonts, and orders this selection in a way meaningful to TEX. Not all the characters from raw fonts need be part of the final virtual font.

Up until virtual fonts, words containing accents suppressed TEX's hyphenation algorithm. We can define an accented letter in a virtual font to be equivalent to any other letter, so hyphenation proceeds unimpeded—yet another advantage of virtual fonts.

To be sure, this unrealistic, alien font provides a contrived example. But with real commercial fonts, these same problems—writ large because real fonts have so many more characters—need the practical solutions that virtual fonts provide.

2 Digital fonts and font tables

Font tables for the Computer Modern TEX fonts and for PostScript outline fonts contain a maximum of 256 positions; 256 is one of the magic numbers of computer science. See figure 1 for examples of real font tables. (For a slightly different representation, refer to the font tables beginning on page 427 of *The TEXbook]/*. The PostScript font tables cannot be shown here; pardon.) A casual glance reveals significant differences between the layouts for the two fonts. Each slot of the Computer Modern font is filled (up till character 127), whereas there are many unfilled slots in the PostScript fonts. Some characters, like the uppercase Greek letters or the ff, ffi, and ffl ligatures, do not appear anywhere in the PostScript font while certain PostScript characters appear nowhere in the Computer Modern layout. Other characters are in disparate positions. All the Scandinavian ligatures (æ, and so on) appear in the fourth row of the Computer Modern font table, but cluster together near the very end of the PostScript font table.

Positions in any font table are numbered starting from zero up to 255. We know that terrestrial TEX selects characters *not* by the character name but according to its position in the font table, so a command to typeset an 'A' is relayed as an instruction to

> typeset the character from the currently selected font that occupies position 65 in that font,

since that's the numeric label of the 'A' slot. (All character positions are given here in decimal notation. Computer scientists may be more comfortable with a character's octal position, which is why that information also appears in the tables.)

When using PostScript outline fonts, it's useful to be able to typeset in a 'fake font'—our virtual font—which looks real to TEX but is in fact an amalgam of one or more raw, component fonts. We arrange this virtual font so the characters in the virtual font are in the same order as in any other Computer Modern font. That way, macros will seldom have to be redefined for different fonts, a particularly important issue for mathematics typesetting.

Associated with a font table is the *font encoding vector* or just the *encoding vector* or *code page*. The encoding vector is the list of the character names in the order in which they occur in the font table. For a TEX font (figure 1), the encoding vector is the list beginning

Gamma, Delta, Theta, Lambda, Epsilon, Pi, Sigma, Upsilon, Phi, Psi, Omega, ff, fi, fl, ffi, ffl, dotlessi, ...

and so on. If we let '`.notdef`' designate a font position for which no character is defined, then for a PostScript font, the encoding vector is a list that begins

.notdef, ..., .notdef (32 times in all),
space, exclam, quotedbl, numbersign, dollar, ...

3 What comprises a virtual font?

TEX does not deal with any characters beyond the metrics associated with a font. It expects to find this information in a `tfm` file, and so each virtual font must be accompanied by a font metric file in the usual way. This file should be placed in a suitable place.

0 Γ 0	1 Δ 1	2 Θ 2	3 Λ 3	4 Ξ 4	5 Π 5	6 Σ 6	7 Υ 7	8 Φ 10	9 Ψ 11	10 Ω 12	11 ff 13	12 fi 14	13 fl 15	14 ffi 16	15 ffl 17
16 ı 20	17 ȷ 21	18 ` 22	19 ´ 23	20 ˘ 24	21 ¯ 25	22 ˚ 26	23 ¸ 27	24 ß 30	25 æ 31	26 œ 32	27 ø 33	28 Æ 34	29 Œ 35	30 Ø 36	31 Å 37
32 ˝ 40	33 ! 41	34 ” 42	35 # 43	36 $ 44	37 % 45	38 & 46	39 ’ 47	40 (50	41) 51	42 * 52	43 + 53	44 , 54	45 - 55	46 . 56	47 / 57
48 0 60	49 1 61	50 2 62	51 3 63	52 4 64	53 5 65	54 6 66	55 7 67	56 8 70	57 9 71	58 : 72	59 ; 73	60 ¡ 74	61 = 75	62 ¿ 76	63 ? 77
64 @ 100	65 A 101	66 B 102	67 C 103	68 D 104	69 E 105	70 F 106	71 G 107	72 H 110	73 I 111	74 J 112	75 K 113	76 L 114	77 M 115	78 N 116	79 O 117
80 P 120	81 Q 121	82 R 122	83 S 123	84 T 124	85 U 125	86 V 126	87 W 127	88 X 130	89 Y 131	90 Z 132	91 [133	92 “ 134	93] 135	94 ^ 136	95 ˙ 137
96 ‘ 140	97 a 141	98 b 142	99 c 143	100 d 144	101 e 145	102 f 146	103 g 147	104 h 150	105 i 151	106 j 152	107 k 153	108 l 154	109 m 155	110 n 156	111 o 157
112 p 160	113 q 161	114 r 162	115 s 163	116 t 164	117 u 165	118 v 166	119 w 167	120 x 170	121 y 171	122 z 172	123 – 173	124 — 174	125 ˝ 175	126 ~ 176	127 ¨ 177

Figure 1: A font table for Computer Modern Roman fonts (here, `cmb10`). The Roman numbers in the upper left of each box give the character number using the usual decimal representation. The italic numbers in the lower right are the octal equivalents.

The details behind the construction of the virtual characters appear in the actual virtual font file, a file with the extension `vf`. There needs to be a place place on a hard disk to store virtual fonts, in the same way that there are places for `tfm` files, format files, input files, and so on. The places have different names depending on whether your system is traditional or complies with the TDS standard (see chapter 6).

The actual virtual font `vf` file contains fragments of `dvi` language that specify the way that a virtual character should be created. That means that a character in a virtual font can be anything that occurs in a `dvi` file. In theory, one virtual character can typeset an entire page or document! Typically, virtual characters are not so complex. In the alien planet example, the virtual font simply remapped characters (placed them in a different and more suitable order) and merged characters together from raw fonts.

4 What we will need; preparation

We've seen in the previous chapter that *vfinst* takes care of the most common virtual font tasks—the installation of scalable fonts to make them usable by TEX. However, there are many more reasons to use virtual fonts, and so we begin a lengthy, conscientious look at virtual fonts. We need to understand, too, the sequence of steps that *vfinst* performs.

Firstly, virtual fonts are a feature of TEX3. In order to proceed, that version must be installed.

Many authors will be preparing documents for output on PostScript printing devices. Since TEX only knows how to write `dvi` files, we will always need a dvi-to-PostScript converter. Frequently these programs require an auxiliary map file to "map" the long font names to the short file names that are all that some operating systems, notably MS-DOS and its relatives, can handle. Because it is freely available, and available for all computer platforms, we will usually refer to *dvips* and its map file `psfonts.map`. Each of its entries pairs a short, DOS-acceptable name for a *raw* font with its long, given font name. These short aliases are the names that we should use in the process of virtual font creation. For each short alias in the map file, there must be a `tfm` file under that name.

The map file may serve other functions. It may aid in the process of downloading (see below), and it may be where we specify certain types of transformations on a font.

At print time, how does the printer get the information about the shapes of the characters in the document? For bitmap fonts, it's the responsibility of the printer driver to include the bitmap information in the instructions it transmits to the printer. For scalable fonts, the situation is different. The outline information on all fonts must be transmitted to the printer, for it is the printer that ultimately converts the outline to raster form for printing. In most PostScript-compatible printers, descriptions of 35 or so common fonts, including Times Roman and Helvetica, are resident—built-in—to the printer. If you use other, non-resident fonts, you will need to *download*—transmit—this font information to the printer, and this downloading can be accomplished in different ways. It is also possible to include the font information in the PostScript version of the document.

5 The purpose of a simple installation

If we examine the font tables in this chapter, we see that the problem of constructing a virtual font from a PostScript font is not hopeless. Most characters are in the same positions, including all upper- and lowercase letters, digits, and much of the punctuation. We may divide the remaining characters in an outline font into two groups:

- special characters like fi, —, ¿, and the American quotation marks "" which are selected by TEX's ligature mechanism; and
- characters like æ, Œ, or ç which are invoked by control sequences or control words (here, `\ae`, `\OE`, and `\c{c}`).

(Actually, there's a third group—those characters present in `cmr10` but absent entirely from a standard Type1 font, such as the ligatures ff, ffi, and ffl. We'll see later how to deal with these.) We would like to make sure we have access to *these* members of a font **without** having to change the rules by which we create our source documents. Actually, just in case an author has been silly and used a nonstandard convention to typeset a symbol (such as getting ¿ by typing `\char62` or `\symbol{62}` rather than `?‘`), we would like the layout of the virtual font to adhere as closely as possible to the original TEX font layout.

The ligatures of the first group can be handled in a nonvirtual way by adjusting the font metric files so TEX plucks the ligature from the proper font position; no remapping is necessary. This requires a modification of the `tfm` only.

Characters accessed by TEX commands present more of a challenge. The definition for each such command relies upon being able to locate special characters by their position in the font table. TEX therefore expects œ to be character 27 in a font, since that's where it is in the Computer Modern family. When constructing ç, it expects the cedilla to be in position 24 for the same reason. Typically, though, these characters do not appear in those positions in the raw PostScript font (œ and cedilla occupy positions 250 and 203). Macros could be redefined, but it's a bad idea to have macro definitions depend on the current font. We require our virtual font utility to reorder—to remap—these characters in the font. For example, virtual character 27 consists of the raw character 250. That way, when the virtual font is the current font, `\oe` will correctly typeset the œ glyph.

The `afm2tfm` utility (part of *dvips*) is an excellent tool for creating this elementary kind of virtual font—a font consisting of the remapping of the characters in a single raw font. Because the source for this program has been made available, `afm2tfm` has been ported to every significant computer architecture, and executable binaries are freely available from friends or software archives (the same applies to *dvips* itself). But `afm2tfm` suffers from several disabilities: it can't create a virtual font out of more than one raw file, it can't create the `fd` font descriptors that LATEX now uses, and it doesn't mimic the original TEX font layout as closely as it might. Nevertheless, simple installations are so common that it is important to detail this process precisely.

The box (not included here—sorry) summarizes the procedure to follow to use `afm2tfm` to create virtual files from outline fonts. We use this procedure whenever this simple manipulation is sufficient for our needs. (More complicated finagling is best carried out with *fontinst*; see below.) This process involves using or creating several file types. If an outline font is `psfont`, that means the distribution diskette should include `psfont.afm` and `psfont.pfb`. It is necessary to rename the file name `psfont8a`, and from these we will be generating files `psfont7t.vpl`, `psfont7t.tfm`, and `psfont7t.vf`, the virtual file. We also generate a font metric file corresponding to the "raw" PostScript file `psfont8a.tfm`.

The program `afm2tfm` can also create pseudo-small caps fonts and other fonts which have undergone simple geometric transformations, like slanting or extension. Check the documentation to learn how.

Once we've created the virtual font and placed all the files where they belong, we access any virtual file just as if it were a normal TEX font (which it is). For example, we could declare

```
\font\foo=psfont7t at 10.5pt
```

in a `plain` TEX document and use it via the command `\foo` which has become a font changing command like `\it` or `\tt`. Although we never again refer to the raw font file explicitly, TEX do. Behind the scenes, whenever a TEX device driver resolves the meaning of a virtual font, it refers to the component raw fonts, the raw fonts must be present on our system.

6 Introduction to *fontinst*

The *fontinst* package, by Alan Jeffrey, does everything `afm2tfm` does and more. It can create a virtual font from several raw fonts, for example, and it automatically produces an auxiliary `fd` file used by LATEX's NFSS to select the font. The *fontinst* package is written entirely in TEX, and TEXegetes will enjoy perusing `fontinst.sty` to watch TEX do things it was never intended for. Writing it in the TEX language insures *fontinst* runs on every platform that TEX does. You can retrieve *fontinst* from any CTAN archive, under `fonts/utils`. The discussion in this chapter supplements `fontinst.tex`, the documentation of the package.

We use *fontinst* by preparing a simple `plain` TEX file. Typically, this file will be short, and will consist of a command to `\input` the `fontinst.sty`, followed by a variety of commands which tell *fontinst* how to create the virtual font. Normally, `vf` and `tfm` files are binary files, file types which TEX cannot write. Therefore, *fontinst* reads and writes property list files and special metric and encoding files instead. These are all in Ascii, and the property files in particular are ASCII equivalents to `vf` and `tfm` files with extensions `vpl` and `pl`. Part of your TEX installation should include the utilities `vptovf` and `pltotf` (together with their inverses `vftovp` and `tftopl`), and we would then use utilities these to create the font files we need.

After each successful run of *fontinst* there will be three new kinds of files in your working directory.

- `pl` files—one for each raw font—which feeds into `pltotf` to create a `tfm` file;
- `vpl` files—one for each virtual font—which feeds into `vptovf` to create one `vf` and one `tfm` file; and
- a `fd` *font descriptor file*—one for each font family—which NFSS will use to relate the font attributes to individual fonts.

(There are also some new `mtx` files and the usual `log` file that you can delete.) It is necessary to run all `vpl` files through `vptovf` and all `pl` files through `pltotf` to generate the binary metric files that TEX needs. A map file, such as `psfonts.map` for *dvips*, must be updated; see chapter 6.

All `tfm` files belong with your other `tfm` files. The `vf` files belong in a special place as well, where *dvips* expects to find virtual files. The `fd` files belong in a TEX inputs directory.

6.1 Installing fontinst

The *fontinst* package consists of a the core file `fontinst.sty` together with some documentation, some samples and many examples. You may well receive the package as a zipped collection of files already organized in its own directory structure. I found it convenient to create a `.../fontinst` directory in which I unpacked *fontinst*. One or two levels down is a new directory called `inputs`. In addition to `fontinst.sty` itself, there are a collection of files with extensions `mtx` and `etx`. Move these files to one of your TEX input directories to complete the installation.

Goals

The *fontinst* package provides a new language for the creation of virtual fonts of all types. Our goal in this and subsequent chapters shall be to develop familiarity with these procedures so we can install any font with (relatively) little work.

Although *fontinst* works *much* slower than `afm2tfm`, it is much faster than creating `vpl` files by hand.

7 Simple font installation with fontinst

7.1 New commands

Figure 2 displays one way to use *fontinst* to install the Times Roman fonts that are resident in every PostScript printer. Most *fontinst* installation files resemble this display.

Much of this file is standard boilerplate. The first line

```
\input fontinst.sty
```

makes *fontinst* known to TEX.

The pair of commands `\installfonts` and `\endinstallfonts` (with no arguments) surrounds the sequence of commands that do the bulk of the work. One or more `\installfamily` commands now follow. The first argument specifies the encoding, the second the family designation, and the third a set of commands that will be executed each time the family is loaded. See the *fontinst* documentation for further details on this third argument; it will be empty in nearly all our work.

```
\installfamily{encoding}{family}
               {fd-commands}
```

The workhorse command in any installation file is the `\installfont` command, which takes eight parameters. The last parameter allows us to specify size information for the font. For scalable fonts, it is nearly always empty because scalable fonts are, well, scalable to any size. (Bitmap fonts, created specifically for different sizes, require non-empty entries.) Parameters 4 through 7 provide space for the encoding, family, series, and shape values that *fontinst* uses to create the NFSS `fd` file. Consult the previous chapter and examples in this and subsequent chapters to see how these parameters fill out. The very first parameter stores the file name of the virtual font you want to create.

That leaves the second and third parameters. In order to understand their significance, we need a small digression to consider the process of font creation.

7.2 Creating fonts

There are two aspects to font creation:

1. **Metric:** We need procedures for constructing each glyph or character in the virtual font.
2. **Encoding:** We need to decide on the order of the glyphs in the font, and specify any additional rules that the characters need to live by. For example, rules might concern ligatures (any time an i follows a single f, replace it by fi; any time an *A* appears at the beginning of a word, replace it by a swash variant), or math symbols (any time interior material gets too tall, replace a delimiter by the next larger size).

For *fontinst*, these instructions should be in *metric files*, with an `mtx` extension, and *encoding files*, with extension `etx`. In the second position of the `\installfont` command, we place a list of metric files to be inserted. *fontinst* reads them to find out how to construct the characters. The third position records the name of an encoding file, which *fontinst* reads to learn which characters to include, how to order them, and what ligature and other special rules to follow.

Schematically, a `\installfont` instruction looks like this.

```
\installfont{font-name}{metric-files}
   {encoding-file}{encoding}
   {family-name}{series}{shape}{size}
```

7.3 Metric files

The task of preparing metric files is lightened because *fontinst* reads three types of metric files:

1. `mtx` files, using a format specific to *fontinst*;
2. `afm` files, the ASCII metric files that come with each scalable outline font; and
3. `pl` files, the ASCII equivalents to a TEX `tfm` file.

fontinst reads the first two types automatically, but you will need to use the program `tftopl` (which should accompany your version of TEX) to create this file. For example, type

```
\input fontinst.sty

\installfonts
   \installfamily{OT1}{ptm}{}
   \installfont{ptmr7t}{ptmr8a,latin}{OT1}{OT1}{ptm}{m}{n}{}
   \installfont{ptmrc7t}{ptmr8a,latin}{OT1c}{OT1}{ptm}{m}{sc}{}
   \installfont{ptmri7t}{ptmri8a,latin}{OT1}{OT1}{ptm}{m}{it}{}
   \installfont{ptmb7t}{ptmb8a,latin}{OT1}{OT1}{ptm}{bx}{n}{}
   \installfont{ptmbc7t}{ptmb8a,latin}{OT1c}{OT1}{ptm}{bx}{sc}{}
   \installfont{ptmbi7t}{ptmbi8a,latin}{OT1}{OT1}{ptm}{bx}{it}{}
\endinstallfonts
\bye
```

Figure 2: One way to install Times Roman. This examples uses the original TEX encoding but does not include any expert fonts. Two series are installed—regular and bold. Within each series, three shapes are installed—upright, small caps (which use encoding file `OT1c.etx`), and italic.

```
tftopl cmr10.tfm cmr10.pl
```

to do the obvious thing.

In *fontinst* prior definitions take precedence over subsequent definitions. That is, if any construct appears more than once in a series of files that *fontinst* reads, only the first one counts; later definitions are silently ignored. Therefore, *the order in which fontinst reads files is critical!* This philosophy is central to the way *fontinst* works, as we'll see.

The file `latin.mtx` is the "metric file of last resort." It provides instructions for creating 401 glyphs found in Latin alphabets. Of those 401, some are unfakable—there's no way to print characters like 'A' unless the A is in the font, but many other glyphs can be faked. Accented letters can be built from letters and accents, and small caps can be taken from an uppercase font set at 80% of the current design size. Of course, there isn't room for all 401 of these characters in a single font anyway. (The limit is 256.) But because many of these characters have been previously defined in metric files, *fontinst* will ignore many of the definitions in `latin.mtx`—remember, glyph constructs have no effect if defined previously. But if you have neglected to define a glyph that you later call for, the definitions in `latin.mtx` serve as safety net. That is why all the `\installfont` commands in figure 2 and in virtually every *fontinst* example contain lists of metric files that terminate with a call to `latin.mtx`.

7.4 Encoding files

Once the metric files have done their job (of constructing the glyphs), a single encoding file chooses the group of characters that belong in the font and the proper order (encoding). This file also specifies certain ligature and other rules for the font to abide by.

Encoding files tend have names to reflect their encoding. Thus, the encoding file for the OT1 encoding is simply `OT1.etx`. Similar files, `OT1c.etx` and `OT19.etx`, would set up a small caps and an old-style figures font using OT1 encoding. There are several more variants in the *fontinst* distribution.

8 Progressive examples

It's time to consider examples using *fontinst* to create virtual fonts.

8.1 Simple font installation

The simplest way to use *fontinst* is to run TEX on the file `fontinst.sty` and to then type

```
\latinfamily{ptm}{}
\bye
```

in response to TEX's star prompt `*`. This works presuming that all the fonts in the `ptm` family (Times Roman) have been named in accordance with Karl Berry's font naming scheme and that all font metric files are in places that TEX can read from.

This method is best for authors who plan never to need any more exotic fonts than these. Subsequent examples are designed to show of the power of *fontinst* and to teach its intricacies in a tutorial manner.

8.2 Easy DC fonts

The Cork encoding, denoted by T1, refers to the standard agreed upon at a TEX meeting held in Cork, Ireland in September 1990. At that time, agreement was reached for sets of 256-character fonts for use by TEX. (The TEX standard had at that time only been extended to 256 character fonts for a short time.) The `dcr` fonts look like the usual computer Modern fonts, but these fonts have been extended according to the Cork standard. Virtual fonts provide an easy way to generate `dcr` fonts from raw, Computer Modern fonts.

For each virtual `dcr` font, a corresponding `cmr` font acts as the single raw font. We will need the property list `pl` file as well.

Here are the steps to create a virtual `dcr10` from a raw `cmr10` font. The installation file `makedcr.tex` should resemble

```
% This is makedcr.tex, for use with fontinst.
\input fontinst.sty
  \installfonts
    \installfamily{T1}{dcr}{}
    \installfont{dcr10}{cmr10,latin}{T1}{T1}%
                {dcr}{m}{n}{}
```

```
  \endinstallfonts
\bye
```

although you'll need additional `\installfont` statements for members of this family which are italic, boldface, and so on.

Following the successful execution of the *fontinst* run, enter these statements at the prompt:

```
tftopl cmr10.tfm cmr10.pl
tex makedcr
vptovf dcr10.vpl dcr10.vf dcr10.tfm
rm *.log *.pl *.vpl *.mtx
```

after which you'll need to move the `tfm` and `vf` files to their proper places. In words, we need first the ASCII property list file, after which we can invoke TEX and *fontinst*. Thereafter, we create binary font files using the virtual property `vpl` produced by *fontinst*. Finally, we clean up. (Unix syntax is shown.) This example does not require an addendum to `psfonts.map` unless you are using scalable versions of the Computer Modern fonts.

Drawbacks of easy dcr10

During the creation of `dcr10.vpl`, *fontinst* reports that 34 glyphs are missing—that is, of the full complement of characters that do belong in a T1-encoded font, *fontinst* complained 34 times that it couldn't make the glyph. All of these are various diacritics (ring, ASCII tilde, and so on) and accented letters that use these missing diacritics, but a few are more problematic, including the sterling symbol and French quotations. If you access these characters, the mock `dcr10` font will not be suitable.

Moreover, there is no premium on disc space from using these fonts. The `vf` and `tfm` files require roughly 4k and 5.7k apiece, comparable with an actual `pk` file at a laser printer resolution.

8.3 Installing outline fonts

The *vfinst* utility takes care of scalable font installation, but we are now in a position to understand a simple installation ourselves. We may begin by renaming the font files to conform to a TEX font naming standard. Suppose we have Adobe Garamond Roman fonts to install. We rename the regular font files to `padr8a.pfb` and `padr8a.afm`, for example.

As an example, we can create the OT1-encoded font `padr7t` from these. This new font will belong to font family `pad` and have NFSS designations of `m` and `n` (medium series, normal shape). With this information, we prepare an installation file that looks like

```
% This is file makepad.tex
\input fontinst.sty

\installfonts
  \installfamily{OT1}{pad}{}
  \installfont{padr7t}{padr8a,latin}{OT1}%
  {OT1}{pad}{m}{n}{}
\endinstallfonts
```

although a real installation will likely contain several `\installfont` commands. The `\installfont` command is quite straightforward. It:

- constructs a font for family `pad`;
- uses glyph information from `padr8a.afm`, and supplements it (if necessary) with instructions from `latin.mtx`;
- applies the OT1 encoding to it; and
- uses the four parameters `OT1`, `pad`, `m`, and `n` for the NFSS `fd` file.

Incorporating expert fonts

For the vast majority of outline fonts, the only way to get the ff, ffi, and ffl ligatures is from an expert font, because these characters are rarely present in a base font. However, `latin.mtx` does create mock characters with these names because slots are provided for these ligatures in the font by the encoding files. Therefore, the way to get the honest double-f ligatures is simply to include the expert font name in the list of metric files in an `\installfont` command. That is, the skeletal installation file listed above would look something like

```
% This is file makepad.tex
\input fontinst.sty

\installfonts
  \installfamily{OT1}{pad}{}
  \installfont{padr7t}{padr8a,padr8x,latin}{OT1}%
    {OT1}{pad}{m}{n}{}
\endinstallfonts
```

Note the presence of `padr8x`, the expert font for Garamond regular.

The box (not included here; apologies) summarizes the bookkeeping involved in completing the installation. this discussion is presented for pedagogical completeness only, for in this case it's better to use `\latinfamily` (see above, section 8.1) or PSNFSS or *vfinst* (refer to chapter 6).

Detailed Contents for "TEX Unbound: Strategies for Font, Graphics, and More

Alan Hoenig

Abstract

This book will be published by Oxford University Press in early 1997. Contact the author at `ajhjj@cunyvm.cuny.edu` for further information.

Detailed Contents

Acknowledgments . . . . . **i**

Introduction . . . . . **iii**

Contents . . . . . **vii**

Detailed Contents . . . . . **ix**

1 About TEX and LATEX **1**

1 An overview of LATEX and TEX 1
Typographic niceties, 2 Scholarly detritus, 3 Why is TEX hard?, 5

2 A brief history of TEX 6
How does one 'TEX' differ from any other?, 7

3 The LATEX life cycle 8
Another perspective, 8

4 A working LATEX system 9
Miscellaneous TEX software tools, 11

5 Getting TEX 12
Unique TEXs, 13 Updating TEX; extending TEX, 14

6 Installing and running TEX 15
The executable, 15 Other files, 16 Initializing TEX with format files, 17 Running TEX, 18

7 Inking the page 20
The characters of a font, 20 Scalable fonts and a PostScript postscript, 21 Phototypesetters and service bureaus, 24 Just for fun, 26

8 Document files 26
Preparation, 28 What do TEX commands look like?, 30 More about TEX's commands, 31 LATEX and TEX , 31

9 Friends of TEX 31
Some popular macro packages, 33 Metafont and MetaPost, 34 The WEB system of structured documentation, 35 Other utilities, 36

10 Learning and joining 36
The public domain, 38 Joining the TEX community, 40

11 Appendix: scalable typography 40

2 TEX, the Internet, and Multimedia **43**

1 Internet resources 43
Archives, 43 Lists and newsgroups, 49 Newsgroups, 53

2 The worldwide Web and hypertext 53
Web locations, 54

3 CDROMs 54

3 Mostly METAFONT **61**

1 Installing METAFONT 62

Types of files, 62 Second, base, 62

2 Running METAFONT 63
Starting and stopping, 66 "Fonts" and fonts, 66 An example font, 67 Customizing the mode, 69 Packing pixels, 71 From Metafont into TEX, 72

3 Some METAFONT conventions 72
Sharped units, 74 Coordinate systems, 75 Prescription versus description, 77 Modes, 78

4 METAFONT as graphics engine? 79
Nice curves, 80

5 Meta-ness 80

6 Computer Modern fonts 84
Parameter files, 86 Driver files, 86 Program files, 87 Custom Computer Modern fonts, 88 Poor person's font sizes, 89 Better fonts at new sizes, 91

7 **PostScript** and METAFONT 92
PostScript as a device driver, 93 Bitmaps into PostScript, 94 Interfacing METAFONT to encapsulated PostScript, 94

8 **MetaPost** 96
The MetaPost cycle, 96 From MetaPost into TEX, 97 Installing MetaPost, 97

9 Other METAFONT work 98

10 Learning more about METAFONT 102

4 Logical Documents via LATEX 105

1 What is logical document structure? 105
General markup, 107 Logical markup in TEX, 107

2 LATEX 108
Environments, 109 Commands and conventions, 109 The preamble, 111 LATEX209, 112 Floats, 112 Other parts of a document, 113 Index preparation, 113 Bibliographies, 116

3 Modifying LATEX 117
The problem, 118 Examine the source, 118 Private command names, 119 Identifying the component, 119

4 Other structured macro packages 121
The synthesis, 121 Extended `plain` macros, 122 The `texinfo` system, 123

5 TEX in the Workplace 127

1 Word processors 127

2 Spreadsheets 130

3 Hypertext 132
Hypertex on the Web, 133 Acrobat, 135 Hypertext archives, 136

4 TEX in science 138

6 Installing and Selecting Fonts 141

1 Preliminaries 141
How should font selection work?, 141 Caveat: why switch fonts explicitly?, 142 Plan of this chapter, 142

2 Naming digital fonts 142
One naming scheme, 143 Naming scalable math fonts, 146 Aliases and alias files, 147

3 Font installation 148
Bitmap fonts, 148 Scalable outline fonts, 148 An overview of PSNFSS, 151 An overview of VFINST, *153*

4 Plain TEX font selection 156
Getting PDCFSEL, *157 Using* PDCFSEL, *157*

5 LATEX's New Font Selection Scheme 161
Font attributes, 162 Using NFSS, 163

6 NFSS: high level commands 163
Commands versus declarations, 164 "Old" LATEX font commands, 164 NFSS commands in context, 164 Typesetting mathematics, 166

7 Mid-level NFSS commands 167

8 NFSS: low level font interface 169

Sizing fonts, 170 *New math fonts, 172* *Math font sizes, 173* *Making fonts visible to NFSS, 174* *Purposes for individual math fonts, 174* *A style or package template, 175* *New versions, 176*

7 Virtual Fonts, Virtual Fonts . . . 179

1 The virtual font concept . . . 180

2 Digital fonts and font tables . . . 183

3 What comprises a virtual font? . . . 186

4 What we will need; preparation . . . 186

5 The purpose of a simple installation . . . 187

6 Introduction to *fontinst* . . . 190
Installing fontinst, 191

7 Simple font installation with fontinst . . . 192
New commands, 192 *Creating fonts, 193* *Metric files, 194* *Encoding files, 195*

8 Progressive examples . . . 195
Easy DC fonts, 195 *Installing outline fonts, 196* *Small caps, 200* *Oblique and unslanted italic, 201*

9 More virtual font projects . . . 204
Adjustments to individual characters, 204 *Adjusting font size, 205* *Oldstyle figures, 206* *Better footnote numbers, 206* *Foreign languages, 207*

10 The File `mergefd.prl` . . . 212

11 Summary of all *fontinst* commands . . . 215

8 Virtual Font Projects . . . 221

1 Getting started . . . 221
Idiosyncratic definitions, 222 *Names of latin glyphs, 222*

2 Underline and strike-out fonts . . . 224
Strikeout fonts, 226 *One-hundred-percent underlining, 228*

3 Poor person's bold fonts . . . 230

4 f-words . . . 233

5 New encodings; alternate fonts . . . 238
Adobe Garamond, 238 *Installation, 243* *Bitstream Bernhard Modern, 243* *New encodings; hidden characters; Mantinia, 256*

6 Two advanced projects . . . 269
Motion picture credits, 269 *Better underlining, 273*

9 More Virtual Fonts . . . 275

1 Letterspacing and tracking . . . 275
A macro approach, 276 *A virtual font approach, 277*

2 Previewing PostScript: hardware strategies . . . 281
Display PostScript, 282

3 Previewing PostScript: software strategies . . . 282
Pixel fonts from outline fonts, 283 *PostScript renderers, 284* *The virtual font way, 287*

4 Proper—optical—sizing of fonts . . . 294
Optical scaling with bitmap fonts, 295 *Optical scaling with "scalable" fonts, 297* *Monotype Times New Roman, 298* *Approximations to optical scaling, 302* *Other optically scaled fonts, 304*

5 Appendix: installing Times New Roman . . . 305

10 New Math Fonts . . . 309

1 Scalable Computer Modern fonts . . . 310

2 Computer Modern math plus new text fonts . . . 310

3 Initial considerations . . . 311

4 The MathInst utility . . . 312
Overview, 312 *Installing MathInst, 314* *Installing the text fonts, 318* *Running MathInst, 318*

5 New math virtual fonts . . . 319

Using the new fonts, 321 The MathTime fonts, 322 The Euler fonts, 323 Lucida New Math, 323 Mathematica math fonts, 327

6 Fine tuning the new math fonts 329
Adding special purpose fonts, 329 Controlling font scaling, 331 Sizing fonts at small sizes, 332 Other adjustments, 332

7 Rogues' gallery 333
Notes on the rogues' gallery, 335

11 Graphic Discussions 369

1 General graphics 370
Without TEX, 370 With TEX, 371 Hybrid approaches, 371 Plan of attack, 372

2 Graphic inclusions 373
Encapsulated PostScript and epsf, 373 Other graphic formats and pbmplus, 379 Including graphics with bm2font, 383

12 Graphics via LATEX and TEX 389

1 Coordinate geometry 389

2 The LATEX extensions 393
Plain pictures, 397 Extensions to `picture`, 397 `picture`-like packages, 398

3 PICTEX 398
Command syntax, 402 PICTEX plus LATEX, 409

13 Using METAFONT and MetaPost 413

1 Basics 413
Basic truths, 414 Variables, 415 File organization, 417 Coordinate systems and points, 418

2 Paths into pictures 419

3 Calculating 421

4 Uses and applications 428
Drawing engines, 430

5 Simple transformations 432

6 Just MetaPost 435
Getting MetaPost, 436 Shades of gray and color, 436 Typesetting a MetaPost graphic, 437 Including text, 438 Drawing graphs, 442 Text in graphics: advanced topics, 446

7 Appendix: summary of MetaPost 453

14 PSTricks 465

1 Essentials 465

2 Getting started 466
Graphic parameters, 466 The `unit` parameter, 468 Other files, 468

3 About PSTricks coordinate systems 469

4 Elementary examples 470
Errors, 470 A PSTricks gallery, 471 Curves, 473 Drawing engine, 474 Clipping, 475 Cusps, 480

5 More advanced examples 484

6 Text connections 491

7 Repetition 496

8 Other tricks 500

9 Appendix: some PSTricks projects 501
Shark, 501 Gradient table, 503

10 Appendix: summary of PSTricks commands and parameters 505

15 MFPIC Pictures 513

1 Getting started 513
How `mfpic` works, 514 Some syntax rules, 514 Prefix macros, 519 Transforming shapes, 520

2 The mfpic process 523
Other frontends to Metafont, 525

3 Appendix: mfpic reference 525
Appendix 1: Basic TEX Commands **531**
1 Basic Keyboard Conventions 532
2 Default Parameters; Parameters and Simple Commands 535
Spaces, 536
3 Simple Page Layout 537
4 Controlling Lines of Text 538
5 Spacing 539
Horizontal Spacing, 540
6 Selecting New Fonts 541
7 Mathematics 544
8 Special Characters 547
9 White Lies; Where to Now? 547
10 Examples: Simple Letters and Reports 548
Appendix 2: More About LATEX **553**
1 Two Flavors of LATEX 553
209 , 553 Enhanced LATEX , 555
2 Getting Classes and Packages 555
3 Page Layout 557
4 Some Environments 558
Letters, 560
5 Document Structure 561
Appendix 3: Producing This Book **565**
Sources and Resources **569**
Index **577**

A LaTeX Tour, part 1: the base distribution

David Carlisle

1 Introduction

In this article[1] I hope to give a 'guided tour' around the files that make up the basic LaTeX distribution. Subsequent articles in this mini-series will cover other packages by the LaTeX development team, and also some of the main contributed packages.

The primary source for LaTeX is the 'CTAN [2]' network of archives, so if I refer to path names of files this relates to the CTAN file structure. Note however that if you obtained LaTeX as part of a 'pre-packaged' TeX distribution, then these files may have been moved (typically documentation files may be separated from TeX source files). Hopefully this will not cause any confusion.

2 The Components of LaTeX

The LaTeX distribution at the CTAN archives is organised into the following directories.

`base` Contains the core LaTeX files. You need at least these files for a minimal LaTeX installation.

`unpacked` Includes *all* the files in `base` together with the result of 'unpacking' the source files. (More about this later.) Thus when obtaining LaTeX you should get either `base` or `unpacked`, but not both. Getting the former saves on time transferring the files, but getting the latter saves time that would be taken unpacking the source files, so which is preferable depends on the relative speed (and cost) of your machine and your connection to the archives.

`packages` Consists of seven independent LaTeX 'extensions' that are written and supported by the LaTeX developers (or the American Mathematical Society).

`amsfonts`, `amslatex`, `babel`, `graphics`, `mfnfss`, `psnfss` and `tools`

These packages will be described in more detail later in the series.

`fonts` The metafont sources and TeX font metric files of a few fonts that LaTeX requires that are not part of the original plain TeX distribution.

`doc` This directory is not part of the main LaTeX distribution, it is generated by the CTAN archives. As a convenience for those people that have not yet installed LaTeX, some of the main introductory documentation files which are available as LaTeX files in the base distribution are made available in this directory as dvi and PostScript files.

`contrib` This directory contains an ever growing number of contributed LaTeX packages, and other extensions, that have been contributed by LaTeX users. They are not part of the 'official' LaTeX distribution, but many of them form a vital part of any 'working' LaTeX installation. The packages are divided into two subdirectories '`supported`' and '`other`', however at the current time one should ignore this distinction when looking for packages to fetch from the archives. Contrary to expectations some of the best supported packages are distributed (at their author's request) from `contrib/other`.

Unfortunately (for mainly historical reasons) people in search of contributed LaTeX packages also need to look in more distant CTAN directories. Firstly, the `macros/latex209/contrib` area on CTAN contains packages that were written for the previous version of LaTeX that has been obsolete since June 1994. Any packages that are still distributed from this LaTeX 2.09 tree are likely to be less than well supported, but you can still find some useful files there. Secondly, there are some packages that work with multiple formats, not just LaTeX, and these are to be found under `macros/generic` or in their own top-level directory, such as `macros/musictex`.

3 Documentation in the Base Distribution

The documentation that comes with LaTeX is of two forms: plain (ascii) text files with extension `.txt`, or LaTeX documents distributed as LaTeX source with extension `.tex`. Generally speaking the text files are mainly of interest to people installing LaTeX, who may need information before they have a working system. Information of more general interest to LaTeX users is normally distributed as a LaTeX document.

3.1 The ASCII text files

3.1.1 Installation instructions

`00readme` Provides a general introduction to the system, and should be the first file to look at when installing LaTeX for the first time.

[1] This is a slightly edited version of an article that first appeared in the UKTUG journal *Baskerville* 6.1.

[2] The Comprehensive TeX Archive Network:
`ftp.tex.ac.uk` (UK), `ftp.dante.de` (Germany), in the USA the mirror `ftp.cdrom.com` of the UK archive should be used in preference to the original CTAN host of `ftp.shsu.edu` which is unfortunately unreliable at the present time.

`install` Provides 'generic' installation instructions, but for many TeX versions more specific instructions that have been contributed by the authors or users of those systems; thus `emtex` gives instructions for the popular emTeX implementation, `web2ctex` gives specific instructions on installing under UNIX , etc.

`tex2` If you have a TeX that pre-dates version 3.0 (which was released in 1989) by far the best thing to do is to update your TeX, but if that is really not possible this file details how LaTeX may be built under TeX2.

`autoload` Describes the installation of an 'autoloading' version of LaTeX. This produces a much smaller format by saving less common commands in external files rather than in memory. These files are automatically 'autoloaded' as required. This version of LaTeX is particularly recommended if you are using a small installation (for instance a 'small' emTeX on a sub-386 PC).

3.1.2 Other text files

`legal` Contains the copyright notices and distribution conditions for LaTeX.

`bugs` Contains instructions on how to compile a bug report (see below).

`patches` Describes the LaTeX patch mechanism that is used for distributing small updates between the 'full' releases. This file also contains a list of all the files that have changed since the last full release.

`changes` A Change Log of all the changes made to the LaTeX files. This is mainly intended for internal use by the LaTeX developers, but some people like to read it.

3.2 The LaTeX 'guides'

These documents are distributed as LaTeX source (i.e., `.tex` files) although as noted in the above introduction, the CTAN archives distribute most of them in ready-formatted versions in the directory `latex/doc` so you can read these before installing LaTeX if you wish. Unlike the ASCII text files described above, most of these documents are primarily intended for *users* of the system rather than system managers and software installers.

`usrguide` *LaTeX2ε for Authors.* This document describes all the main new features of the 2e release of LaTeX. It was written originally with the user of the old LaTeX 2.09 in mind, but newcomers to LaTeX who have never used the old version should still gain something by reading this document. It does not however cover the majority of LaTeX commands that were not changed, and so it is not a substitute for a full LaTeX manual.

`clsguide` *LaTeX2ε for class and package writers.* A companion to to `usrguide`, gives details of the LaTeX commands for structuring class files and extension packages.

`fntguide` *LaTeX2ε font selection.* For font addicts only, but if you want to know the detailed specification of the 'New Font Selection Scheme' commands, here is the place to look.

`cfgguide` *Configuration options for LaTeX2ε.* Discusses what you can (and can not) do to configure a LaTeX installation to the requirements of your local site.

`ltx3info` *The LaTeX3 Project.* A brief summary of the aims of the LaTeX3 project, the group of volunteers that has taken on the maintenance and development of LaTeX.

`modguide` *Modifying LaTeX.* This document discusses some of the rationale behind the LaTeX distribution conditions as expressed in `legal.txt` and `cfgguide.tex`. Unless you are making a distribution of a modified version of LaTeX, or are particularly interested in software copyright issues, you probably do not want to read this.

3.3 LaTeX News

As well as these larger documents there are a series of one-page 'newsletters'. A new one is produced with each full release of LaTeX. These detail any changes that have occurred in LaTeX or the main extension packages over the six months since the previous release. (LaTeX releases occur at regular intervals, in June and December of each year.) Currently the four files `ltnews01`–`ltnews04` are distributed; these correspond to the four releases of LaTeX since June 1994.

3.4 Example Documents

There are two (very) small example documents, as described in the LaTeX book by Leslie Lamport.

`small2e` A very small (1 page) LaTeX document.

`sample2e` A slightly larger document.

3.5 Documented sources

The source for the LaTeX format, and for all the packages and classes in the core distribution is distributed as '`dtx`' files. These are LaTeX documents which may be processed in the usual way to produce typeset documentation. For example a command such as `latex ltpictur.dtx` would produce documented source of the picture mode commands. The files with names of the form '`lt...dtx`' make up the source of the LaTeX format. If you want to produce a combined document incorporating all these files, you may process `source2e.tex`. This document will produce a typeset version of the LaTeX sources, together with change log and index. It is well over 500 pages long, and so may take a long time to produce. It may produce an index that is too large to be handled by the 'makeindex' program on smaller machines.

3.6 Errata

The principal documentation for LaTeX is the two books *LaTeX: A Document Preparation System*, and *The LaTeX Companion*. Errata for these (and the German edi-

tion of *The Companion*) are available as `manual.err`, `compan.err` and `begleit.err`.

4 The LaTeX Bug Report Database

As described in the file `bugs.txt` mentioned above, the LaTeX3 project maintain a database of bug reports for LaTeX.

If, after checking with colleagues, reading the manual, etc., you decide that some behaviour of LaTeX is incorrect then you may send a message to the LaTeX bug database. Before doing this you should check that your LaTeX is not more than one year old (the bug may have been fixed in a recent release). If you have access to the World Wide Web, you may access the database and see if the problem is already reported by using the search page accessible from: `http://www.tex.ac.uk/CTAN/latex/bugs.html`.

If you decide to send a report, two files are available to help compose a message in the correct format:

`latexbug.tex` LaTeX this file and you will be prompted for information such as your name, and the name of a test file that shows the problem. A mail message will be written to the file `latexbug.msg` which should be sent to `latex-bugs@uni-mainz.de`. (You should *always* use `latexbug.tex` to generate messages to be sent to this bug address. It is an interface to a database (the GNU GNATS problem tracking system) and can not handle messages that are not in the special format written by `latexbug.tex`.

`latexbug.el` For users of the GNU Emacs text editor, a more convenient interface is provided by this file. It runs `latexbug.tex` automatically, and provides online help for filling in the various fields, and finally automatically mails the message to the correct address.

5 Docstrip files

As mentioned above, LaTeX is distributed as documented sources. The files that are actually used by TeX are extracted from these files by running `docstrip.tex`. The LaTeX distribution contains many files with extension `.ins` that control how `docstrip` extracts each file. Most of these are never used individually, as they would just 'unpack' one small part of the distribution. The file `unpack.ins` is a 'master' installation script that calls the smaller install files in turn and so unpacks the whole distribution. Normally running TeX on this file is the first step in installing LaTeX. This step may be omitted however if the `unpacked` directory is obtained from CTAN rather than `base`. `unpacked` is *exactly* the result of obtaining `base` and running TeX on `unpack.ins`. If you have a slow machine you may prefer this route as it saves unpacking time, but conversely it requires downloading more files, so if you are transferring the files via a slow connection such as a modem then you may prefer to get the smaller 'base' distribution.

There are three install files that are *not* included into `unpack.ins` so you may have need to run these if you need the following features.

`autoload` Processing `autoload.ins` will generate the source file for the 'autoload' version of LaTeX, `latexa.ltx`, as described in `autoload.txt`. This should be processed with iniTeX to create a format file to be used in place of the standard `latex.fmt`. As well as the modified format, various packages are created containing the code that has been taken out of the format. Normally these do not need to be invoked explicitly as they are loaded on demand when they are needed. Currently the following package files are produced.

- `autopict` Source for picture mode.
- `autotabg` Source for tabbing environment.
- `autoerr` The texts of most LaTeX error commands.
- `autofss1` Less used font selection commands.
- `autoout1` Code related to `\enlargethispage`.

The autoload format is still quite experimental, and so the range of such 'autoloading' packages may change with future releases.

`cmextra` Processing `cmextra.ins` installs the '`fd`' files for the 'concrete' variants of the Computer Modern fonts, and also the AMS Cyrillic fonts.

`olddc` If using the Computer Modern fonts in the 8-bit 'T1' encoding, LaTeX defaults to using the 'dc fonts'. During 1995 these fonts were updated and the names of the fonts *changed*. Thus the 10 pt roman font corresponding to `cmr10` is now `dcr1000` rather than `dcr10`. The install file `unpack.ins` includes `newdc.ins` so by default LaTeX will use the new 1995 names (dc fonts release 1.2 or later) when using T1 font encoding. If you still have the old dc fonts, then you must process `olddc.ins` to produce suitable `fd` files referring to the old names.

6 The Standard LaTeX Classes

The general appearance of a LaTeX document, and the specification of the commands available is specified in a *document class*. This may be further modified by loading *packages*, as described in `usrguide`. In this section I give a brief overview of the available classes in the base distribution. They all have extension `.cls` (after being unpacked from the `.dtx` source file during the installation process).

`article` 'Article Class'. In some sense the canonical reference class against which all others are judged. This class (which is generated from the same `classes.dtx` source as report and book described below) is a mixed blessing. On one hand

it provides quite a rich collection of commands for marking up documents that means that it serves well as as the basic 'generic' class to be used when no more suitable specific class is available. On the other hand the visual appearance of documents produced with this class is very distinctive. Many people who say they "don't like LaTeX" and so use some other format such as plain, in fact are misled into believing that LaTeX *is* this class. In fact by loading article and then making small adjustments one can produce very different visual designs. The class files for *Baskerville* and TUGBOAT are examples of such non-standard classes based on article.

However for many purposes, portability is more important than original typographical design, and in these cases the article class has the big advantage of being installed at all LaTeX sites.

`report` 'Report Class'. Very similar to article (and produced from the same source). The main differences are that this class has a higher level of sectioning command (`\chapter`) than is available in article, and the front matter is typeset differently.

`book` The book class is again very similar to report with the addition of a few extra features for controlling the front matter and back matter. It is unlikely that you would want to use this class 'as is' for a book, as you would almost certainly want to spend some effort (and perhaps money!) on an original design. However it can be used as a basis or example of the implementation of a LaTeX class for book production.

`letter` This provides commands for producing one or more letters. Many sites use this as a basis for producing site-specific letter class files, for instance with a modified heading that inserts a departmental logo and address.[3]

`proc` Proceedings class. This is a variant of article class (and inputs the `article.cls` file when used). It defaults to two column mode and makes one or two other small adjustments. It may be used as a model for how make a class that builds on another.

`slides` The slides class. This class essentially provides the functionality that was formally built into SLiTeX. It provides a mechanism for producing pages suitable for projecting on an overhead projector. It is described in the LaTeX book, and some people like it; however if you are making a lot of such presentations you may prefer to look at the contributed classes seminar (T. v. Zandt) or foiltex (J. Hafner). These provide alternatives to the standard class that many people find more useful.

As well as these 'Standard Classes' the base distribution contains a few other special purpose classes.

`minimal` This is the minimal LaTeX class. It just sets up a text area, and a font in a single size. None of the normal sectioning or font size commands are available. This class is not intended to be used in documents, but it is often useful when testing macros as it loads very quickly.

`ltxguide` A special purpose class for the 'LaTeX guides' mentioned earlier.

`ltnews` The class file used for the 'LaTeX News' news sheets.

`ltxdoc` This class is used in all the `dtx` documentation files. It is based on the article class and the doc package, but with additional commands for documenting the LaTeX sources. It was not conceived as a class for general use, but some people find it convenient to use it when documenting their own package files.

7 Standard Packages

7.1 Encoding Packages

One of the main features of the 2e release of LaTeX is that it attempts to remove all 'hard wired' assumptions about the encodings being used, both for input and also in the fonts used for typesetting.

It maintains a strict distinction between the *Input Encoding* and the *Output Encoding*. The input encoding relates to the text that you type; this may be a standard encoding such as ASCII (the traditional 7-bit encoding) or ISO -latin-1, or a platform-specific encoding such as 'Windows ANSI ' as used on MicroSoft Windows 3.x machines. The output encoding for text fonts is usually either OT1 (the encoding devised by Knuth and implemented in the original Computer Modern TeX fonts.) or T1 (the new TeX encoding also known as 'Cork' after the meeting where it was agreed).

LaTeX maintains this separation by *always* translating input to an *Internal Encoding*. This is essentially traditional TeX 7-bit input. This internal encoding is then translated to the encoding used in the font without reference to the original input mechanism used. Thus if you specify an input encoding that includes the character **é** you may type that directly at the keyboard, and see it as a single character; however internally LaTeX will treat this as `\'{e}`. If you are using 7-bit OT1 encoded fonts this command will use the `\accent` primitive to add an acute to the **e**; however if you are using T1 fonts, the existing **é** will be accessed directly. Note however that the position of **é** in the output encoding (T1) is typically *different* from the position of the character in the input encoding used.

`inputenc` Specifies that an 8-bit input encoding is being used. A package option should always be used which sets up the default encoding. The currently available options include latin1, latin2, ansinew, cp437, cp437de, applemac. (The two IBM codepage 437 variants differ in just one slot, the former uses β, the latter uses ß.)

[3] One should be able to find details of such local variants in the famous 'local guide'.

So typical usage (to specify ISO Latin-1 input conventions) would be:
`\usepackage[latin1]{inputenc}`

`fontenc` Specifies the default output encoding for text fonts. Currently the available options are OT1 and T1. So to specify that fonts in the the T1 (Cork) encoding be used in the document one would declare:
`\usepackage[T1]{fontenc}`

7.2 Remaining Packages in the Base Distribution

`alltt` Defines the alltt environment, similar to verbatim except that `\`, `{` and `}` retain their usual TeX meanings.

`doc` The package defining the commands used for documenting all the LaTeX code in the distribution.

`shortvrb` This package (really a small part of the doc package) defines the `\MakeShortVerb` command that allows shorthands like `|\foo|` instead of `\verb|\foo|` This is very convenient if you are documenting TeX or some other situation where you need to make a lot of use of short sections of verbatim text.

`exscale` For mainly historical reasons LaTeX always uses the math extension font (used for brackets and sum and integral signs etc.) at the same size, whatever the current font size. This package modifies this behaviour so that magnified fonts are used at larger sizes. At the same time it makes the plain TeX commands `\big`, `\bigg` etc., work as expected in conjunction with LaTeX size commands.

`flafter` LaTeX floats such as the gure and table environment can float *up* to the top of the current page. This means that it is possible that the figure appears before its first reference. Some publisher's styles do not allow this. after redefines the float placement algorithm so that a float never appears before its position in the source file, so by using this package, and placing the gure environment after the first reference to the figure, one can ensure that a figure will appear after the reference.

`graphpap` The `\graphpaper` command produces a grid for use in the picture environment.

`ifthen` Provides an 'if ... then ... else...' programming construct for use in LaTeX packages. Many of the examples in *The LaTeX Companion* assume this package has been loaded.

`makeindx` Implements support for generating an index.

`pict2e` This package produces an error message to say that it has not been written. The documentation in the `pict2e.dtx` source file suggests alternative packages for extending picture mode, pspicture (D. P. Carlisle), pspic (K. K. Thorup), and the very powerful pstricks (T. V. Zandt).

`showidx` This causes the argument of each `\index` command to be printed on the page where it occurs. See also `idx.tex` described below.

`syntonly` Used to process a document without typesetting it. On some systems this speeds things up considerably, and so may (possibly) be useful while debugging documents.

`tracefnt` This allows you to control how much information about LaTeX's font loading is displayed.

`latexsym` Loads the special LaTeX symbol font and then defines commands such as `\Box` that use this font. These commands were defined by default in LaTeX 2.09.

`newlfont` Defines 'old' font commands to act in the 'new' way. For example it makes `\rm` essentially equivalent to `\rmfamily`. This package is not now recommended but is distributed so old documents written using the LaTeX 2.09 version of this package still work.

`oldlfont` A companion to newlfont. This package is only to be used for old documents that used the LaTeX 2.09 package of the same name.

8 Font Definition Files

Unpacking the LaTeX distribution creates dozens of 'font definition files' with extension '`.fd`' from their documented sources (with extension `.fdd`). These map the internal LaTeX model of fonts on to the external file names as used on your system. Normally you never need to load these explicitly into a LaTeX document and they will not be considered in detail here except to say that if you obtain some new fonts from the TeX archives, make sure to also get the related `fd` files, and install them where LaTeX can find them.

9 Makeindex Styles

The distribution includes three styles (with extension `.ist`) for the *makeindex* index generator. They modify the makeindex defaults so as to work with the special requirements of the doc package.

`gind` Produces indices of command definition and use.

`gglo` Produces 'change log' entries (using the LaTeX `\glossary` command rather than `\index`).

`source2e` This style is produced only if the LaTeX document `source2e.tex` is processed. It is almost identical to `gind.ist` but defines 'I' to be in the series 'I–J–K' rather than 'I–II–III'. This is needed for the numbering conventions used in that document.

10 Miscellaneous Utilities and Files

`idx.tex` Print out index entries in your document.

`lablst.tex` Generate list of labels used in a document. You may prefer instead to have the labels show up in the margins of your drafts, in which case use the showkeys package from the 'tools' collection to be described later in this 'tour'.

`ltxcheck.tex` This 'document' should always be processed after LaTeX has been installed. It pro-

duces no output but checks that various components of the system are configured correctly for your machine type.

`nfssfont.tex` Test file for testing a font. A more extensive font test is available if you use the fontsmpl package from the 'tools' collection.

`testpage.tex` Test file for checking the accuracy of a printer. This is particularly useful to see if you need to specify any offsets to your printer driver to ensure that the printed text is correctly positioned on the paper.

`Makefile.unx` A very simplistic template 'Makefile' for installing the LaTeX base distribution under UNIX . Many UNIX TeX distributions come with far more suitable installation procedures. For example the excellent 'teTeX' distribution allows you to install TeX, LaTeX, metafont, dvips, xdvi, and a host of other utilities and fonts just by typing `sh install.sh`.

`latex209.def` This file is loaded whenever a document begining with `\documentstyle` is seen. It forces LaTeX into '2.09 compatibility mode' which is a slow, but a fairly accurate, emulation of the old version of LaTeX. This enables old documents to be processed under the current system.

11 Conclusion

The base distribution described here is only really a bare minimum LaTeX installation. Most sites would want to install all of the LaTeX files from the CTAN `latex/packages` directory, and some of the contributed packages as well. These other files will be covered in future installments of this 'tour'. Part 2 will cover the tools and graphics distributions. Part 3 will cover babel, mfnfss and psnfss. Part 3 will cover the AMS LaTeX distributions amsfonts and amslatex. If I am still feeling keen, later parts may venture out to describe some of the more popular contributed packages.

A LaTeX Tour, part 2: the Tools and Graphics distributions

David Carlisle

1 Introduction

In the previous article in this series I started by giving a description of the files in the 'base' LaTeX distribution. In part 2, I shall cover the 'tools' and 'graphics' distributions. These are distributed in the `tools` and `graphics` subdirectories of the CTAN directory `macros/latex/packages`. Although these files are not part of the minimal base distribution they should normally be included in the LaTeX installation at any site. The LaTeX book assumes that at least the graphics distribution is installed.

The primary source for LaTeX is the 'CTAN [1]' network of archives, so if I refer to path names of files this relates to the CTAN file structure. Note however that if you obtained LaTeX as part of a 'pre-packaged' TeX distribution, then these files may have been moved (typically documentation files may be separated from TeX source files). Hopefully this will not cause any confusion.

2 The Tools Distribution

The tools distribution consists of packages written by individual members of the LaTeX3 project. They are supported by the same mechanism as the base LaTeX distribution, that is any problems should be reported using `latexbug.tex` and the LaTeX bug report database, as described in part 1. Note that this bug report system should *not* be used for 'contributed' packages that one may find in the `macros/latex/contrib` area of the CTAN archives.

2.1 Packages Extending the array and tabular Environments

The first group of packages extend the functionality of the standard LaTeX array and tabular environments. These are all described in Chapter 5 of *The LaTeX Companion*, as well of course as in the source '`.dtx`' files which may be processed by LaTeX to produce typeset documentation, and optionally code listings.

`array` Extended versions of the array, tabular and tabular* environments. The principal advantage of the versions provided by this package is that you can specify typesetting instructions to apply to a whole column of the table. As well as the usual `clr` column specifiers, one may add commands at the beginning of each entry with `>` and at the end of each entry with `<`. So a column specifier of `>{\bfseries}c` would produce a bold, centred column of a table.

The array package also provides a `\newcolumntype` command for defining new column specifiers, in addition to the standard ones. This is used by some of the packages described below.

`dcolumn` Alignment on 'decimal points' in tabular entries. Requires array. This package provides a new column specifier `D` which may be used to produce columns of numbers aligned on a decimal point '.' or some other symbol, such as '·' or ','.

`delarray` This package requires the array package. It provides a mechanism for specifying 'large delimiters' around arrays. This is most convenient for putting brackets around arrays that are to be aligned on their top or bottom row (when the 'obvious' construction with `\left` and `\right` does not work). Compare the standard

```
\left(
  \begin{array}[t]{c}a\\b\end{array}
\right)
\left(
  \begin{array}{cc}a&b\end{array}
\right)
```

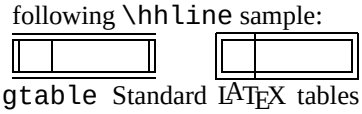
$$\left(\begin{array}{c}a\\b\end{array}\right)\left(\begin{array}{cc}a&b\end{array}\right)$$

with the effect produced using the dcolumn syntax.

```
\begin{array}[t]({c})a\\b\end{array}
\begin{array}({cc})a&b\end{array}
```

$$\begin{array}{c}a\\b\end{array}\begin{array}{cc}a&b\end{array}$$

`hhline` Finer control over horizontal rules in tables. Requires array. Standard LaTeX's `\hline\hline` construction produces a double rule across a table, however the user has no control over how this rule interacts with vertical rules. Using the `\hhline` command provided by this package, one can make 'corners' where a double horizontal rule meets a double vertical rule, and other similar effects.
Compare the first, standard construction with the following `\hhline` sample:



`longtable` Standard LaTeX tables (i.e., the tabular environment) produce 'boxes' that can not be broken across a page. This has advantages in that

[1] `ftp.tex.ac.uk` in the UK

the table can then be positioned just like a large 'character' for instance centred by the center environment, but has the disadvantage that large tables need to be broken by hand to fit on the page. The longtable environment is essentially the same as tabular but produces tables that break at page boundaries, and has some additional commands to control 'head' and 'foot' lines of the table that are added to each page. If the array is also loaded, then the extra features may also be used in longtable column specifications. Note that longtable can deal with *very* long tables, longer than can be stored in memory by TeX's primitive `\hline` command. The longtable package has a few quirks and features that mean that it is not suitable in all cases. An alternative package (currently maintained by Johannes Braams, but as a contributed package, not as part of the tools distribution) is the package supertab which provides a similar supertabular environment.

`tabularx` Defines the tabularx environment which is similar to tabular* but modifies column widths, not inter-column space, to achieve a desired table width.

One common request is to combine the features of tabularx with longtable, i.e., have a table across multiple pages, in which the widths of the 'parbox' columns are calculated automatically. This functionality is not provided by the standard packages in the tools distribution, but the experimental contributed package ltxtable does provide such an environment. (ltxtable is written by the same author as longtable and tabularx; however, problems with ltxtable should *not* be addressed to the LaTeX bugs system.) An alternative to ltxtable is Anil Goel's contributed package, ltablex, a similar merger which is simpler to use than ltxtable, but not quite as powerful.

2.2 Missing File Error Control Files

Although these files (which are all generated from the same source file, `fileerr.dtx`) are distributed as part of the LaTeX distribution, they are possibly of more use when used with *other* formats. The primitive TeX behaviour if asked to input a non-existent file is to offer a prompt:

```
Please type another input file name:
```

You *must* type a valid file name to this prompt or TeX just repeats the request. On some systems you can use a mechanism to abort the job (e.g., control-C or control-Z) but there is no way to tell TeX to skip the input or do any other error recovery.

To avoid this unpleasant loop, LaTeX always checks that a file exists before trying to input it (unless you use the primitive `\input filename` syntax with no `{ }` around the argument).

If you do encounter this loop using a different format, or with LaTeX, by mistyping the file name on the command line, then the following `.tex` files provide valid filenames that you can easily remember which you can type to the missing file prompt. The actions that each of these `.tex` files takes is designed to mimic the actions that are possible after a TeX error.

`h` Typing `h` ⟨*return*⟩ to the missing file prompt will cause TeX to input `h.tex`, this produces a helpful message, and then produces the normal 'error prompt' i.e., `?` so you can hit ⟨*return*⟩ to move on, or `x` to quit, or whatever.

`s` Typing `s` ⟨*return*⟩ inputs `s.tex` which puts TeX into 'scroll mode'. This means that it will scroll past future errors without stopping.

`x` Typing `x` ⟨*return*⟩ causes the current TeX run to be aborted.

`e` The file `e.tex` is in fact the same as `x.tex` but allows `e` to be given as an answer to the missing file prompt similar to the `e` response to the error prompt (which is supposed to start up an editor but usually is the same as `x`).

`␣` If the operating system allows there will also be a file `.tex` which does nothing, this will mean that just hitting ⟨*return*⟩ in response to the missing file prompt inputs `.tex` and allows TeX to proceed with the original file. Some operating systems object to a file with only an extension and no filename before the '`.`' so this option may not be available to you. Most TeX distributions include a file `null.tex` which is also empty, so if you do not have the option of installing the file `.tex` you may type `null` ⟨*return*⟩ in response to the missing file prompt, which will also allow TeX to proceed.

2.3 Miscellaneous Tools Packages

`afterpage` Defines an `\afterpage` command that saves up its argument and executes it after the current page (i.e., at the top of the next page). LaTeX's output routine was *not* designed with the idea that packages might want to play kind of trick, so this package is particularly fragile. In fact it was only written as a kind of private joke; I noticed the comment "*Output routines are always protected by enclosing them in groups, so that they do not inadvertently mess up the rest of TeX*" in the TeXbook, and wanted to answer[2] the question, "*Where do you end up if you jump out of that group with* `\aftergroup`*?*" Despite this, judging by comments in `comp.text.tex`, people do seem to find the package useful...

`enumerate` Extended version of the enumerate environment. The environment is given an optional argument which controls how the counter is printed.

[2] The answer incidentally is not at the top of the next page, but rather any of the places where the TeXbook uses the magic phrase "*exercises the page builder*".

For example `\begin{enumerate}[a)]` would produce items labelled 'a)' ' b)' ' c)'.

`ftnright` Place footnotes in the right hand column in two-column mode. Normally LaTeX places footnotes at the bottom of each column. This package causes the footnotes for both columns of a page to be set in the normal text area at the end of the second column on each page. It currently works only with the standard two column mechanism, not with the mechanism of the multicol package.

`indentfirst` Indent the first paragraph of sections etc. This very small package just suppresses the usual LaTeX mechanism which ensures that the first paragraph of each section is not indented.

`multicol` Typeset text in columns (up to 10 columns per page), with the length of the final columns 'balanced'. *Baskerville* uses this package to balance the columns at the end of every article. Unlike the standard `\twocolumn` command, this package allows changing the number of columns part way down a page. It does have some restrictions on the use of floats which means that it is not suitable for all purposes. Also (uniquely for the files in the core LaTeX distribution) this package has special conditions on commercial use.

`rawfonts` Preload fonts under the old internal font names of LaTeX 2.09. Not recommended for new packages, but may help when updating old files.

`somedefs` This package is not intended to be called directly by a document, but may by used (via `\RequirePackage`) to build a package in which you want the default behaviour to be to execute *all* possible options, but that the user may execute just some of the options by specifying options in the `\usepackage` call. This is used in the rawfonts package above to allow just some of the 'old' font names to be defined rather than all of them.

`theorem` Flexible definition of 'theorem-like' environments. The standard `\newtheorem` command gives some control over the title and numbering of 'theorem-like' declarations, but is not very flexible. The theorem package provides an enhanced declaration scheme which gives control over the fonts used in the heading and theorem body, and such details as whether the numbering is '**Theorem 1**' or '**1 Theorem**'. Recently this package has acquired a close cousin, the amsthm package, part of the 'AMS-LaTeX' collection. The AMS variant has perhaps slightly simpler user-syntax but is used in much the same way.

`varioref` Provides `\vref` and related commands. `\vref` is like a combination of `\ref` and `\pageref` which produces references such as 'Figure 2 on page 3'. However, it omits the page number if it is on the current page, and replaces it by phrases such as 'on the facing page' when appropriate.

`verbatim` Flexible version of verbatim environment. The standard LaTeX verbatim environment can not easily be used in the definition of other environments as typically the `\end` of the newly defined environment is not recognised as such, but is treated as verbatim text. This package re-implements verbatim such that (with some restrictions) it can be used in the definition of other environments and commands. It also defines some such derived environments, for inputting and writing files verbatim, and for adding line numbers, and also a comment environment that ignores all the environment body.

`xr` The xr (external references) package allows one LaTeX document to access the `.aux` file of another. So if file `fileA` has a section marked with `\label{xyz}` then file `fileB` may refer to that section using `\ref{xyz}` just as if it were part of the same document. This requires the file `fileA.aux` created when `fileA` was processed to be still available when `fileB` is processed. (This package was originally by Jean-Pierre Drucbert, but was recoded and adopted into the tools distribution.)

`xspace` One of the more common errors in TeXdocuments is to use a command such as `\TeX` within text, but forget to follow it with `\␣` or `{}`. This package defines a command `\xspace` which may be used at the end of the definition of such a 'text command'. It looks ahead at the next token and adds a space unless that token is a punctuation character.

2.4 Packages for Drafts and Tests

`fontsmpl` Package and test file for producing 'font samples'. The base distribution contains a file `nfssfont.tex` that shows some small samples, and a character table for a given font. `fontsmpl.tex` produces a much more extensive test showing examples of all the fonts in a given family. If you want to devise your own similar test suite you may use the fontsmpl package, following the examples in `fontsmpl.tex`.

`layout` Defines a `\layout` command that produces a half size 'picture' of the document page settings such as `\textwidth`, `\oddsidemargin`, ... together with a table of their values. This is quite useful when designing a new class file, as it gives a visual representation of how the various areas of the page for headlines, body text, marginal notes, etc., relate to each other.

`showkeys` LaTeX's automatic numbering and cross referencing feature is one of its strongest points, as it makes editing a document (and thus potentially changing the numbering throughout the rest of the file) quite painless. However, one disadvantage is that when reading a printed draft, one sees 'final' numbers rather than the symbolic names that are used in the source file's `\label` and `\ref` command. This package makes these symbolic names, or 'keys', appear in the margin in the case of

lbl

`\label` and `\bibitem` or raised above the number, like this [tbt], in the case of `\ref` and `\cite`. Some people find the raised labels above cross references distracting and so a package option turns them off, just leaving the marginal notes showing the `\label` and `\bibitem` keys.

3 The Graphics Distribution

TeX (and the `dvi` format) is only designed to deal with rectangular boxes consisting of text, white space or rectangular rules. However it has an 'escape mechanism', the `\special` primitive command that allows processing instructions to be passed straight from TeX (via the `dvi` file) to the 'driver' program that is used to process (e.g., preview or print) the `dvi` file. TeX places essentially no restrictions on what instructions may be passed via `\special`, and so the possibilities are unlimited...

Most modern drivers can import 'graphic' files of various sorts. Those drivers that are producing PostScript can often do more extensive manipulations of the typeset text, such as scaling or rotation of the text, or even writing text along an arbitrary curve. Many of these drivers can also support colour to some extent. Unfortunately as all these features require that the `dvi` file stores processing instructions for the driver, it means that the `dvi` file is not portable to a site that uses a different driver program. There have been many attempts over the years to coordinate the `\special` syntax used by the different drivers, so that they would all accept a common core of processing instructions, but there has been notable lack of success in such efforts to date...

As a 'next best thing' to having portability at the `dvi` level, LaTeX supplies a suite of standard graphics commands provided by the packages described in this section, so that at least TeX source files should be reasonably portable. At a given site the graphics packages will be customised to use a suitable 'back end' file that converts the LaTeX syntax into the form required by the local driver. This should mean that as long as both drivers support some feature, such as including PostScript graphics, a file just needs to be re-processed with LaTeX to use the `\special`s at the new site; the LaTeX file does not need to be edited. Although this suite of programs was devised as part of LaTeX, users of other TeX formats may use them by way of the interface available from CTAN hosts in `macros/generic/graphics`.

3.1 Documentation

All the packages in this distribution are, as usual, distributed as documented sources in `dtx` form, however the documentation in these package sources is rather technical. A separate 'User Guide' is available as LaTeX source in `grfguide.tex` and also in pre-formatted form in the PostScript file `grfguide.ps`.

The color package (which produces colours despite the strange spelling) and the graphics package are also described in Lamport's LaTeX manual. An alternative to `grfguide.tex` as a free source of documentation is Keith Reckdahl's *Using EPS Graphics in LaTeX2ε Documents* distributed from CTAN sites in the file `info/epslatex.ps` and published in TUGBOAT 17, no. 1–2. This document covers the graphicx package in some depth, and also related contributed packages for controlling figure placement and captions, and the psfrag system for overlaying LaTeX text over a PostScript diagram.

3.2 Colour

`color` Produce coloured effects in your document. The `\color{red}` declaration would make all the following text red, the similar `\textcolor` command takes an extra argument that specifies the text to be coloured (by analogy with `\rmfamily` and `\textrm`).
One may also produce boxes with coloured backgrounds using the `\colorbox` command.
Accurate treatment of colour is probably the feature that requires the most 'help' from the driver program. If your driver was not specifically written to support colour then probably the color package will not work at all, or will be limited to regions of colour that fall on one page, and all the current colours will be 'forgotten' at a page break.

`pstcol` The pstricks package of Tim van Zandt provides a very powerful interface to PostScript. Unfortunately the package has some slight incompatibilities with the color package. If a document loads this pstcol package, both color and pstricks are loaded, and then a few internal pstricks functions are redefined to repair the incompatibility.

3.3 Rotation, Scaling and Graphics Inclusion

`graphics` This is the core LaTeX package for rotation (`\rotatebox`), scaling (`\scalebox` and `\resizebox`) of text, and the inclusion of graphics images (`\includegraphics`). Unlike the old LaTeX 2.09 packages such as ps g the `\includegraphics` command is not restricted to PostScript graphics, but can include any graphics formats that your driver supports.

`graphicx` Often when including graphics files one needs to specify combinations of scaling and rotation and other special effects. The graphics package uses standard LaTeX 'positional' optional arguments which means that it is not practical for any command to support more than a couple of optional arguments. The graphicx package calls the graphics package internally, but offers a more powerful and friendly 'named argument' interface in which an arbitrary number of optional keys may be set in one `[ ]` argument. For instance to include a graphic scaled to half size, and rotated through 90°, one can specify

```
\includegraphics[scale=.5, angle=90]{file}
```

To do the equivalent with the graphics package would require nested calls of

`\includegraphics` inside `\scalebox` inside `\rotatebox`.

`lscape` Provides a landscape environment within which the body of every page is rotated through 90°. The page head and foot are not rotated, but stay in their usual positions. It requires the graphics package which is used to handle the rotation.

`epsfig` The obsolete LaTeX 2.09 did not come with a standard graphics package. Two popular contributed packages to include PostScript graphics were ps g (T. Darrell) and epsf (T. Rokicki). Sebastian Rahtz merged and extended these to produce the package eps g. The eps g package became very popular, especially after it was given extensive coverage in *The LaTeX Companion*. For this reason the current distribution contains a package called eps g so that old documents do not need converting to the new system. However this eps g is just a wrapper that converts the old syntax into calls to the new `\includegraphics` command and so should not now be used for new documents.

3.4 Driver Files

As mentioned above these packages all require customisation to a particular driver. This may be specified either in a site configuration file, or as a package option in the document. The code for these drivers is all stored in '`.def`' files, so for instance the code for the *dvips* driver (and also for *xdvi* which uses the same `\special` syntax) is stored in `dvips.def`. All these driver files are derived from the same source, `drivers.dtx`, except for the Textures file which is currently distributed as a separate `textures.def` contributed by Arthur Ogawa. One special `.def` file does not correspond to a driver, `dvipsnam.def` predefines the 60+ colours that are 'known' to the *dvips* driver. It may be used with other drivers as well, as described in the color package documentation.

3.5 Other Graphics Packages

The remaining two packages do not have code that is specific to dvi driver programs, and so in some sense do not really belong in the graphics distribution; however they are used by the graphics and graphicx packages. In fact the code of either of these packages may be extracted and used in any format based on plain TeX. They do not use any LaTeX specific features.

`keyval` The graphicx package makes use of a 'named argument' or 'key equals value' syntax as described above. Keyval provides a general parser for such a syntax, so this package is unlikely to be directly called within a document, but may be loaded by `\RequirePackage` by any package or class file that needs to define commands with such a syntax.

`trig` This package provides functions for calculating the trigonometric functions sin, cos and tan. These are used by the graphics package for determining the amount of space a rotated box will take up.

4 Coming Soon

Part 3 of this tour will describe the files of Johannes Braams' babel distribution of packages for multi-lingual typesetting, the psnfss distribution of PostScript font related packages, and mfnfss distribution of packages for loading 'Pandora' and 'Old German' fonts.

FRISTI

The Foundation for Responsible Info Stuffing Inventions

Herman Haverkort

hh@fgbbs.iaf.nl

September 1996

Abstract

This article gives an impression of my way of designing and typesetting very small telephone and address lists, birthday calendars etc. Design considerations and typesetting tricks are presented. The latter include macros which define the sheets (from A4 to credit card and even key fob size), macros to process data records (typically specified in a separate file) in various ways, and macros to stuff and stow data in very limited space.

Keywords: small, list, directory, calendar, key fob, credit card, stuffing

Preface

The macros which are presented in this article are not treated in full detail, some are only vaguely mentioned. This article is not meant to be a manual, my goal is to give you an impression of what can be accomplished with LaTeX in the field of Responsible Info Stuffing Inventions. All macros presented in this article are part of the hhfristi package. If you are interested in getting the macros and more thorough explanation, I will happily provide them on request.

The development of all that is presented here, originated in the shortcomings of an ordinary membership list of an extraordinary youth orchestra. The orchestra consists of about fifty members, which are listed yearly on an A4 sized address list, which is handed out to all members. I found that the address lists were ill suited to quite a big part of their daily use, and I wanted to make better lists. Moreover, I wanted to typeset them automatically. The design principles presented in this article are mostly based on the practical problems I encountered; they are not the result of a thorough study of lists in general and therefore may not always be as valuable as they are to me.

Choosing the size of the list

An A4 size list is easy to produce, since it is the standard paper size for typewriters, copying machines etc.[1], and it is very well suited to be stored in an archive, which typically is an ordner, or a box, containing a lot more A4's which the list can be stored nicely in between. However, an A4 size list may not be such a good list to have at hand in varying circumstances. To stuff an A4 in a wallet, you must fold it at least three times. On the folds the text will wear off, and eventually the list will tear. Besides, an A4 size sheet of paper folded three times will be almost one millimeter thick; having a few of those lists in your wallet will contribute quite a lot to uncomfortable thickness. Furthermore, big lists, especially wide lists, are often hard to read correctly, as will be explained further on.

For always-at-hand lists I prefer a page size of A7 maximum. One can construct bigger lists, but then I recommend doing it in such a way that the list can be folded to A7 size or less, without having the folds cut the text. In table 2 you will find a summary of the list sizes supported by the hhfristi package.

The hhfristi list sizes and shapes can be selected using the `fristiform` environment. Text inside this environment is formatted to fit the specified shape and size, and consequently output in one or two boxes (for one and two sided lists). These boxes can be printed next to each other by including the `fristiform` environment in a `\hbox`, or on top of each other by using a `\vbox`. The `fristiform` environment requires one argument which is a list of options separated by commas. Simple options consist of a single keyword; value options consist of a keyword followed by the '`=`' character and some kind of value. The options supported are listed in tables 1 and 2.

Selecting font sizes

To make it easier to experiment with different font sizes I developed a macro `\hhfrsizes`. A typical use of this macro reads `\hhfrsizes{8pt}{1.05}`. It sets `\normalsize` to 8pt, sets all other font size macros (`\scriptsize`, `\small`, `\large` etc.) to appropriate values relative to the `\normalsize`, and sets the baseline stretch to 1.05. All font sizes calculated are rounded to standard values between 5pt and 24.88pt (10pt $\times 1.2^5$).

[1] At least it is in the Netherlands; substitute a similar size for other countries in this sentence.

Simple options		
keyword, selected by default	**alternative option**	**description**
`landscape`	`portrait`	to be used after size or width and height options; landscape has no effect; portrait switches width and height values
`row`	`stack`	determines if pages are to be arranged as a row (vertical folds and turn-over axis) or as a stack (horizontal folds and turn-over axis)
`noeye`		specifies that no space for a perforation is to be created
`nooutline`	`outline`	determines wether or not the outline of the list shape is to be drawn

Table 1: Simple options for the fristiform environment

Value options			
keyword	**values**	**default**	**decription**
`size`	see descr.	A6	legal values are for example: **C** credit card size (85.5mm × 53.8mm); **A6** A0 size, halved six times by shortening its longest side (148.7mm × 105.2mm); **A5/2** A5 size, halved by shortening its *shortest* side (210.5mm × 74.3mm); **A4/2/3** A4 size, with its shortest side divided by two and its longest side divided by three (105.2mm × 99.1mm); Please note that any positive integers can be used instead of the values used above; any basic size identifier can be used instead of the 'A' and 'C' used above. Size identifiers 'A', 'B', 'C', 'E' (executive paper) and 'L' (letter paper) are predefined with macro calls like `\hhfr@basicsize {A}{1189.21mm}{841.90mm}`; other identifiers can be defined similarly; the size options always sets width and height so that the width is the longest side
`width`	dimensions	148.7mm	sets the total width of the list
`height`	dimensions	105.2mm	sets the total height of the list
`columns`	1, 2, ...	1	selects number of columns per page
`colsep`	dimensions	4pt	sets distance between columns
`colseprule`	dimensions	0.4pt	sets width of rule between columns
`folds`	0, 1, ...	0	selects how many folds the list should have
`fold`	dimensions	15pt	sets the amount of unused space around folds
`hem`	dimensions	6pt	if the number of folds is two or more, then strips parallel to the folds are created at the ends of the list; the width of the strips is two times the hem value (the strips can be doubled up to provide a kind of 'handles' which can be gripped to unfold the list)
`sides`	1, 2	2	selects one or two sided list
`outerhmar`	dimensions	6pt	sets the horizontal distance between border and text (outer left and right margin)
`innerhmar`	dimensions	6pt	sets the horizontal distance between fold space and text (left and right margin around folds; the total horizontal distance between two pages equals the fold plus two times the innerhmar)
`outervmar`	dimensions	3pt	anologous to outerhmar
`innervmar`	dimensions	3pt	anologous to innerhmar
`eyepos`	l, r, t, b	not set	determines where space for a perforation should be created
`eye`	dimensions	20pt	sets the diameter of the space taken by the perforation (this space is independent of the perforation outline which is drawn)
`eyemar`	dimensions	0pt	sets the extra margin between the perforation space and the text (this margin is added to the outerhmar or outervmar used)
`corner`	dimensions	10pt	sets the radius of the rounded corners
`seal`	dimensions	0pt	the seal value represents the extra margin for sealing the laminate; it is subtracted twice from the total width and height
`fontsize`	dimensions	not set	sets fristi font sizes to the specified value and sets baselinestretch to one using `\hhfrsizes`
`stretch`	positive v.	not set	sets the baselinestretch to the specified value

Table 2: Value options for the fristiform environment

Designing the lay-out of a list

While lay-outing an address list, it is important to realize how the list will be used. A typical list contains one line for every person who is on it, while each line contains a number of fields of information about a person. Information can be read vertically (e.g. reading the residence field of every person) or horizontally (e.g. reading all the information about one person). In general only the names (family or first names, depending on the application) are read vertically, so names should be listed on a vertical line, below each other. Thus it is easy to read vertically and to use the alphabetical order to find a particular name. When that particular name is found, information about that person is read horizontally. To ease this horizontal reading it is important that the information about one person is on a horizontal line which the eye can easily follow. In table 3 this is clearly not the case. The horizontal lines are unclear, so that it is hardly possible to find correctly someone's telephone number without using a ruler or moving your finger horizontally along the paper, from name to number.

In table 4 the horizontal lines are much clearer because all information about one person is listed directly next to each other, without large white gaps in between. Of course vertical reading of other fields than the name is very difficult on this list (which is irrelevant in many practical applications), and the table does not look very neat.

Table 5 shows a compromise which looks better to some folks, and enables vertical reading of all fields. The horizontal lines are elucidated by connecting rules between the fields, and by additional separating rules after each fifth line.

In general shorter horizontal lines give better readability (this is also a reason why big lists are not always better than small ones). In table 6 this is reckoned with in various ways, although this list is also a compromise between various readability and aesthetic demands. In this case the information which is needed most is the telephone number; therefore it is typeset close to the name. Next is the postcode, which I often do not know by heart, and finally the address, which is put at the end because I often do know it by heart. Extraordinary long names, like Victoria van Asch van Wijck, are typeset (partly) in a smaller font. This injures the readability of that particular name, but if I had enlarged the gap between name and phone number on all other lines instead, I would have injured the readability of the whole list. Above that I would be forced to create space by choosing a smaller (less readable) font for the whole list.

Stuffing techniques

When making the list in table 6, I intended to stuff a lot of information on a small sheet of paper in a very well readable fashion, so: in a font size as large as possible, using abbreviations only when needed. I used the following tricks to stuff info:

- printing extraordinary long names in a smaller font;
- printing information which is less frequently needed in a smaller font;
- raising abbreviated prefixes etc. (thus the period and the space which normally follow an abbreviation can be omitted);
- using different fonts for different fields (thus making it possible to put different fields very close to each other, without making the optical distinction between them unclear; see, for example, the postcodes and phone numbers);
- omitting the zero (the 'interlocal access code') that precedes the area codes;
- omitting area codes which equal that of most members, leaving more space for printing the names;
- omitting the first digit of the postcode if it equals that of most members (in this case: all members);
- omitting residences (which I often know by heart, and if not, they can be deduced from the postcodes using the table which is printed on the head of the list), unless there is enough space left in the address field;

Of course, the last three tricks can only be applied for local societies etc.

Most of the above mentioned tricks I only want to use when compact typesetting is really necessary. Therefore I designed 'stuffers', macros which can act differently depending on the stuffing pressure. An example of such a macro is `\sml`. The macro `\sml` is defined using `\defstuffer {sml}{3}{#1#2#3}[2]{{\small #2}}`: `\sml` gets `{3}` arguments, `\sml` normally typesets them all (`{#1#2#3}`), but if the level of stuffing pressure is `[2]` or less (the lower the number, the higher the pressure), only the middle argument is typeset, and it is typeset small. Stuffing directions like `\lng`, `\sht` and `\abb` are used to choose between full typesetting and abbreviations, and in the case of residence names: between typesetting them and entering them in the postcode table.

In general stuffers are passive: they always act as if there were a lot of space. However, when using the macro `\hhfrsqueeze` they are useful. `\hhfrsqueeze {`*dimension*`}{`*text*`}` tries to stuff *text* in a `\hbox` which is at most *dimension* wide. It does so by experimenting with different settings of the level of stuffing pressure, so that stuffers contained in the *text* can have its use.

If the option `dutch` is used, macros for "van de", "straat" etc. are predefined, making use of appropriate stuffers.

I did not design a general framework for handling text which really does not fit yet. However I did handle a special case: the case of two items sharing partly the same information, for example two persons having the same address. A macro is provided which facilitates trying to typeset them both on one line, and if that turns out to be impossible, typesetting them each on a separate line after all. I will not treat this `\hhfrduo` macro in detail here because until now the only practical examples of its use are quite complicated.

Arts, Lieke	Gemertstraat 16	6844 HC	Arnhem	(026) 39 11 1 11
van Asch van Wijck, Victoria	Tunnelweg 4	6601 CW	Wijchen	(024) 66 16 9 57
Bezemer, Sem	Alkmaarsingel 230	6843 WR	Arnhem	(026) 6 84 628 6
Buddeke, Sarah	Sint Caeciliapad 35	6815 GM	Arnhem	(026) 48 40 17 2
van den Bijlaard, Hans	Laan van Klarenbeek 105	6824 JN	Arnhem	(026) 41 26 20 5
van den Bijlaard, Quirijn	West Breukelderweg 15	6721 MP	Bennekom	(0318) 1 21 75 6
Constandse, Arthur	Wielewaalstraat 5	6823 DA	Arnhem	(026) 9 48 92 99
Ehlert, Arvid	Gemertstraat 17	6844 HD	Arnhem	(026) 7 85 05 67
van Elden, Ariette	Prins Bernhardweg 18	6862 ZH	Oosterbeek	(026) 36 44 34 2
van Elden, Remelie	Prins Bernhardweg 18	6862 ZH	Oosterbeek	(026) 36 44 34 2
van Es, Xander	Haarlemweg 20	6843 AM	Arnhem	(026) 40 13 85 4
Fabels, Marc	Rijnkade 39c	6811 HA	Arnhem	(026) 4 90 4 008
van Ge en, Pim	Zwarteweg 3	6923 CK	Groessen	(0316) 7 27 9 39
Geurts, Karin	Mierlostraat 92	6844 DZ	Arnhem	(026) 41 16 30 0
van Gurp, Thomas	Annastraat 7	6862 CG	Oosterbeek	(026) 38 37 35 6
Haverkort, Herman	Zijpendaalseweg 17	6814 CB	Arnhem	(026) 35 16 7 23
Haverkort, Koen	Monnikensteeg 264	6823 AL	Arnhem	(026) 51 39 36 1

Table 3: An example of a list with unclear horizontal lines

Arts, Lieke Gemertstraat 16 6844 HC Arnhem (026) 39 11 1 11
van Asch van Wijck, Victoria Tunnelweg 4 6601 CW Wijchen (024) 66 16 9 57
Bezemer, Sem Alkmaarsingel 230 6843 WR Arnhem (026) 6 84 628 6
Buddeke, Sarah Sint Caeciliapad 35 6815 GM Arnhem (026) 48 40 17 2
van den Bijlaard, Hans Laan van Klarenbeek 105 6824 JN Arnhem (026) 41 26 20 5
van den Bijlaard, Quirijn West Breukelderweg 15 6721 MP Bennekom (0318) 1 21 75 6
Constandse, Arthur Wielewaalstraat 5 6823 DA Arnhem (026) 9 48 92 99
Ehlert, Arvid Gemertstraat 17 6844 HD Arnhem (026) 7 85 05 67
van Elden, Ariette Prins Bernhardweg 18 6862 ZH Oosterbeek (026) 36 44 34 2
van Elden, Remelie Prins Bernhardweg 18 6862 ZH Oosterbeek (026) 36 44 34 2
van Es, Xander Haarlemweg 20 6843 AM Arnhem (026) 40 13 85 4

Table 4: An example of a list with unclear vertical lines

Arts, Lieke — Gemertstraat 16 — 6844 HC Arnhem — (026) 39 11 1 11
van Asch van Wijck, Victoria — Tunnelweg 4 — 6601 CW Wijchen — (024) 66 16 9 57
Bezemer, Sem — Alkmaarsingel 230 — 6843 WR Arnhem — (026) 6 84 628 6
Buddeke, Sarah — Sint Caeciliapad 35 — 6815 GM Arnhem — (026) 48 40 17 2
van den Bijlaard, Hans — Laan van Klarenbeek 105 — 6824 JN Arnhem — (026) 41 26 20 5

van den Bijlaard, Quirijn — West Breukelderweg 15 — 6721 MP Bennekom — (0318) 1 21 75 6
Constandse, Arthur — Wielewaalstraat 5 — 6823 DA Arnhem — (026) 9 48 92 99
Ehlert, Arvid — Gemertstraat 17 — 6844 HD Arnhem — (026) 7 85 05 67
van Elden, Ariette — Prins Bernhardweg 18 — 6862 ZH Oosterbeek — (026) 36 44 34 2
van Elden, Remelie — Prins Bernhardweg 18 — 6862 ZH Oosterbeek — (026) 36 44 34 2

van Es, Xander — Haarlemweg 20 — 6843 AM Arnhem — (026) 40 13 85 4
Fabels, Marc — Rijnkade 39c — 6811 HA Arnhem — (026) 4 90 4 008
van Ge en, Pim — Zwarteweg 3 — 6923 CK Groessen — (0316) 7 27 9 39
Geurts, Karin — Mierlostraat 92 — 6844 DZ Arnhem — (026) 41 16 30 0

Table 5: An example of vertical lay-out with elucidated horizontal lines

Top level TEX code:

```
\begin{fristimax}{AIO \today}{%
  size=A6,portrait,stack,%
  folds=1,fold=7mm,outervmar=4mm,%
  innervmar=0mm,outerhmar=2mm,
  corner=0mm,outline,%
  addresswidth=.425\hsize,%
  fontsize=12pt,stretch=.89,%
  textshape=\sffamily,%
  numbershape=\rmfamily\bfseries,%
  telarea=26,postarea=6}%
  \input ledenlst.dat
\end{fristimax}
```

AIO 26 september 1996 Netnrs. 26, tenzij anders vermeld. Postcodes beginnen met 6.
Woonplaatsen: (6)51–(6)54 Nijmegen (6)721 Bennekom (6)81–(6)84 Arnhem
(6)861–(6)862 Oosterbeek (6)866 Heelsum (6)871 Renkum (6)874 Wolfheze
(6)88 Velp (6)923 Groessen (6)93 Westervoort © FrisTi

Lieke Arts ——— **39 11 1 11** *844 HC Gemertstraat 16*
Victoria vAschvWijck **24-66 16 9 57** *601 CW Tunnelweg 4 Wijchen*
Sem Bezemer ——— **6 84 628 6** *843 WR Alkmaarsingel 230*

Sarah Buddeke —— **48 40 17 2** *815 GM Sint Caeciliapad 35*
Hans vdnBijlaard —— **41 26 20 5** *824 JN Ln vKlarenbeek 105*
Quirijn vdnBijlaard - **318-1 21 75 6** *721 MP W. Breukelderwg 15*

Arthur Constandse – **9 48 92 99** *823 DA Wielewaalstraat 5*
Arvid Ehlert ——— **7 85 05 67** *844 HD Gemertstraat 17*
Ariette, Remelie vElden **36 44 34 2** *862 ZH Pr Bernhardwg 18*

Xander van Es —— **40 13 85 4** *843 AM Haarlemweg 20*

Marc Fabels ——— **4 90 4 008** *811 HA Rijnkade 39c Arnhem*
Pim van Ge en –**316-7 27 9 39** *923 CK Zwarteweg 3 Groess.*

Karin Geurts ——— **41 16 30 0** *844 DZ Mierlostraat 92 Arnh.*
(e) loes.vd.tuin@tip.nl
Thomas van Gurp — **38 37 35 6** *862 CG Annastraat 7 O'beek*
Herman Haverkort – **35 16 7 23** *814 CB Zijpendaalseweg 17*
(e) hh@fgbbs.iaf.nl

Koen Haverkort —— **51 39 36 1** *823 AL Monnikensteeg 264*
Hester Hendriks —— **41 49 20 7** *862 CG Annastraat 27*
Ellen Hiemstra — **317-9 24 25 5** *866 ET Schutterspad 31*

Annebrecht 't Hoen - **3 62 47 42** *881 JN Schpsdr Overbeek 3*
Nanda van Huet —— **3 25 47 68** *852 MD Dullert 26 Huissen*
Tjeerd Kalsbeek —— **3 61 471 3** *826 JC V Kinsbergstr 46*

Table 6: Example of a fristimax list, based on the input of which some lines are presented in table 9 (only one side shown)

To conclude the section about stuffing techniques I will give an overview of some formatting utilities defined in hhfristi:

\hhfrtel gets four arguments: a one argument macro, the area code prefix, the area code postfix, and a phone number. The macro is used to typeset the whole: it is typically a font selection macro like `\textbf`. The prefix is typically "(0", the postfix ") ", and the phone number is specified by giving numbers, separated by dots. The first dot specifies the end of the area code, the following dots specify places where thin spaces should be inserted to group the digits. The area code, including prefix and postfix, is omitted if it equals the default area code defined by `\hhfr@telarea`.

\hhfrlocalpostcode gets three arguments: a macro to typeset the digits, a macro to typeset the alphabetic characters, and the postcode itself. In hhfristi a version for numeric postcodes and a version for Dutch postcodes are defined. The first digits or characters of the postcode are surrounded by `\default{` and `}` if they equal the default leading characters defined by `\hhfr@postarea`. By letting `\default \relax` or `\@gobble` one can choose if default characters should be typeset.

\hhfrrecordpostcodes gets three arguments. The third specifies a residence name, the first specifies its lowest postcode (or the lowest leading discriminating digits), the second its highest postcode. `\hhfrrecordpostcodes` inserts the residence name with its postcodes in the postcode list, unless it is already there.

\hhfrpostcodes gets two arguments: a title (for example: "Residences" and a macro to typeset postcodes. It typesets the postcode list filled by `\hhfrrecordpostcodes`.

\hhfrgobblecentury gets four arguments; a typical use is `\hhfrgobblecentury 19'\year`. If

the expansion of `\year` starts with “19”, the first two digits are omitted and replaced by the apostrophe. `\year` will be fully typeset otherwise.

Data entry

I wanted to be able to specify the data to be listed in such a way that it would be independent of the lay-out. After all, I wanted to be able to use the same data file for very different lists, which do not only differ in the selection of data and the order in which the fields have to be presented, but also in the stuffing tricks used, and in the way in which the stuffing mechanism takes advantage of the structure of the data. For example: in some lists two persons living at the same address are put together on one line, in some lists they get two successive lines, and in other lists they would be typeset far apart from each other because the whole list would be sorted by first name instead of last name.

To facilitate all this I decided to enter the data in the following way. In principle all information about (for example) a person is given by listing pairs of tags and values. Tag and value are separated by a colon; pairs are separated by semicolons and spaces; the entire record is embraced. If multiple persons share the same information, the shared information is specified at the top level like described above. Then a list of persons is inserted instead of the remaining tag and value pairs. The list simply consists of a sequence of embraced subrecords. The same approach is taken if one person has multiple addresses, or telephone numbers, or whatever. Then the person’s name can be considered shared information for the phone numbers, and the phone numbers are put in a list. In table 7 examples are given of ‘singles’, persons sharing last name and address, persons sharing the address only, and even multiple persons, sharing the same address and sharing last names in groups.

The input shown in table 7 is part of the input for the credit card size horse owners and address list, of which both sides are shown in table 8, together with its TEX code. The hard work is done by the `fristidata` environment. `fristidata` gets two arguments: an initialisation sequence, and a list of patterns. The initialisation sequence, `\gdef\namen{}` in this example, is executed just before handling each data record. The list of patterns specifies what to do with each record. `fristidata` tries to match patterns and data from the input file. Whenever a match is found, the macros specified in the pattern are called and the next record is read from the file. For example: the first pattern matches data containing one value for the “prd” tag, one or no value for “str”, “hnr”, “pcd”, “wpl” and “tel”, one value for “fam”, and one or more values for “naam”. First, for each “naam” value, the macro `\voornaam` is called. Finally, the macro `\totaal` is called for handling the entire record.

If no pattern matches the data given, `fristidata` tries to restructure the data to fit a pattern. For example, the complex structure of names in the Goddijn-van-Weelden-family does not match any pattern. Therefore the structure of the data is simplified automatically by `fristidata` by ‘distributing’ the last names over the family members.

The macros which handle the actual processing of the data use some macros presented in the foregoing. A few are new:

`\hhfrqueue` gets three arguments: a macro name, a delimiter, and some text. It extends the definition of the specified macro by appending the text. If the macro was not empty the delimiter is used. For example, suppose that `\namen` has been defined by `\gdef\namen{}`. Then `\hhfrqueue\namen{ en }{Frans}` defines `\namen` to expand to “Frans”, and a following `\hhfrqueue\namen{ en }{Herman}` redefines `\namen` to “Frans en Herman”.

`\frat` gets one argument, a data tag. It expands to the value paired with the tag in the record concerned.

`\withfrat` gets two arguments, a data tag and some text. It expands to the text only if there is some value attached to the tag. For example: `\withfrat{hnr}{~\frat{hnr}}` is effective only if some value for “hnr” (the house number) is known, and if so, the number is typeset prefaced by a tie.

In table 9 you will find a few example lines from a data file which combines data tags and stuffing information. The stuffing information does not influence the typesetting of the data directly: it only indicates some stuffing *possibilities.* Wether or not these possibilities are used indeed is a matter of lay-out, which is not specified in the data file. The file regio.dat, which is input, defines macros with the names and postcodes of all cities and villages within about twenty kilometers from Arnhem. Street names often contain macros like `\weg` or `\straat`, which expand to stuffers which specify how to print those words in full and how to print them abbreviated.

```
{naam:Priscilla; fam:Beijkirch; tel:26.3.21.01.71;
  pcd:6852gn; str:Holthuizerdreef; hnr:360; wpl:Arnhem;
  prd:Camachio}%
{{naam:Sanne}{naam:Rens}; fam:Wallenburg; tel:26.3.25.88.72;
  pcd:6852jh; str:Orionsingel; hnr:5; wpl:Huissen;
  prd:Cinderella}%
{{naam:Janneke; fam:van Kleef}{naam:Frans; fam:Agterberg}; tel:26.3.25.24.08;
  pcd:6852ml; str:Perenbongerd; hnr:5; wpl:Huissen;
  prd:Palomino (J\&F's Exclusive)}%
{{{naam:Frans}{naam:Lore}; fam:Goddijn}%
  {{naam:Jacoline}{naam:Veerle}; fam:van Weelden}; tel:26.3.21.93.42;
  pcd:6832dd; str:Bereklauwstraat; hnr:63; wpl:Arnhem;
  prd:Kasper}%
{{naam:Theo; fam:Cillessen; tel:26.3.81.29.97;
  pcd:6843bw; str:Medemblikhof; hnr:12; wpl:Arnhem}%
  {naam:Monique; fam:Geerlings; tel:481.37.37.38;
  pcd:6661gc; str:Keijserstraat; hnr:13; wpl:Elst};
  prd:Sultan}%
{naam:Jan; fam:Nas; tel:481.46.22.60;
  pcd:6681ja; str:De Pollenbrink; hnr:2--4; wpl:Bemmel;
  dienst:Veearts}%
```

Table 7: A part of the input for the card shown in table 8.

```
\def\voornaam{\hhfrqueue\namen{, }{\frat{naam}}}
\def\helenaam{\voornaam\hhfrqueue\namen~{\frat{fam}}}
\def\naamenadres{\hhfrqueue\namen{\hfill\protect\newline}{%
  \frat{naam}~\frat{fam}\quad\adres}}
\def\adres{\frat{str}~\frat{hnr}\nobreak\quad
  \protect\hhfrlocalpostcode{\textnr}{\uppercase}{\frat{pcd}}~\frat{wpl}\quad
  \withfrat{tel}{\protect\hhfrtel{\textnr}{(0}{)\,}{\frat{tel}}}}
\def\totaal{\raggedright\emergencystretch=.9\hsize\relax\hangindent1em
  \withfrat{prd}{\textbf{\frat{prd}}: }%
  \withfrat{dienst}{\textsl{\frat{dienst}}: }%
  \namen
  \withfrat{fam}{~\frat{fam}}%
  \withfrat{str}{\quad\adres}\par}
\begin{fristiform}{size=C,outline,seal=10pt,outervmar=6pt}
  \hhfrsizes\@vpt1\sffamily\parskip\z@\parindent\z@
  \noindent\hbox to \hsize{\textbf{Holthuizen \today}\hfil}
  \par\addvspace\baselineskip
  \begin{fristidata}{\gdef\namen{}}{%
    +prd ?str ?hnr ?pcd ?wpl ?tel +fam *1{+naam -> \voornaam} -> \totaal;
    +dienst ?str ?hnr ?pcd ?wpl ?tel +fam *1{+naam -> \voornaam} -> \totaal;
    +prd ?str ?hnr ?pcd ?wpl ?tel *2{+fam +naam -> \helenaam} -> \totaal;
    +prd *2{?str ?hnr ?pcd ?wpl ?tel +fam +naam -> \naamenadres} -> \totaal}
    \input{paarden.dat}
  \end{fristidata}
  \fristifoot
\end{fristiform}
```

Holthuizen 26 september 1996

Camachio: Priscilla Beijkirch Holthuizerdreef 360 6852 GN Arnhem (026) 3 21 01 71
Cinderella: Sanne, Rens Wallenburg Orionsingel 5 6852 JH Huissen (026) 3 25 88 72
Canoura: Astrid Verhoef Vlet 31 6852 DL Huissen (026) 3 25 02 29
Durbin: Christa Volmanbeck Hazelaarstraat 29 6841 AD Arnhem (026) 3 21 08 64
Dusty: Diana Willemsen Holthuizerdreef 41 6852 JH Huissen (026) 3 25 01 53
Palomino (J&F's Exclusive): Janneke van Kleef, Frans Agterberg
Perenbongerd 5 6852 ML Huissen (026) 3 25 24 08
Finesse, La: Rachel Stenger Parkdreef 22 6852 BG Huissen (026) 3 25 36 45
Fire: Rebecca Verhoef Vlet 31 6852 DL Huissen (026) 3 25 02 29
Flame, Red: Eva Corton Scheprad 18 6852 BT Huissen (026) 3 25 65 09
Flash: Priscilla Bos Kuunskop 50-A 6852 JT Huissen (026) 3 25 54 82
Fleurie (J&F's -): Janneke van Kleef, Frans Agterberg Parkdreef 5 6852 ML Huissen (026) 3 25 42 08
Flip: Ilene Hattink Bastion 6-A 6852 CW Huissen (026) 3 25 84 93
Floortje: Jantine Busscher Loostraat 114 6852 BD Huissen (026) 3 25 63 18
Greetje: Melissa Janssen Siriusdreef 5 6832 GT Arnhem (026) 3 21 60 51
Ingmar: Debbie de Wilde Beemd 105 6852 MH Huissen (026) 3 25 69 37
Jolina: Guuske Busscher Loostraat 114 6852 BD Huissen (026) 3 25 63 18

Jolly Jumper: Mariska Neijenhuis Kuunskop 16 6852 JT Huissen (026) 3 25 76 34
Joury: Denise Vriends Kolk 70 6852 KB Huissen (026) 3 25 01 09
Kasper: Frans Goddijn, Lore Goddijn, Jacoline van Weelden, Veerle van Weelden
Bereklauwstraat 63 6832 DD Arnhem (026) 3 21 93 42
Lady Faradiba: Ramon Damen Endepoel 34 6852 LG Huissen (026) 3 25 78 54
Eefje Hendriks Stationsstraat 12-A 3811 MJ Amersfoort (033) 46 3 46 92
Midnight: Manon Huisman Kersenbongerd 5 6852 BJ Huissen (026) 3 25 22 61
Morris: Suzanne Dijkstra Rijkenstraat 19 6851 ME Huissen (026) 3 25 20 58
Sultan: Theo Cillessen Medemblikhof 12 6843 BW Arnhem (026) 3 81 29 97
Monique Geerlings Keijserstraat 13 6661 GC Elst (0481) 37 37 38
Whisky: Peter Hollander De Loohof 126 6671 AV Zetten (0488) 45 32 57

Kippen: G.J. Demon M.L. Kingstraat 12 6852 AV Bemmel (026) 3 25 41 13
Veearts: Jan Nas De Pollenbrink 2{4 6681 JA Bemmel (0481) 46 22 60
Zaak: Henk van Kleef K. Lantermansplein 6 6671 ZH Zetten (0488) 45 21 04

© FrisTi

Table 8: A credit card size list of horse owners and addresses, with the TEX code (two sides shown).

```
\input regio.dat
\def\Wijchen{\res{6601}{6605}{W"ych.}{W"ychen}}

\itm
{naam:Lieke; geb:15/3/1977; fam:Arts; tel:26.39.11.1.11;
  pcd:6844hc; str:Gemert\straat; hnr:16; wpl:\Arnhem}%
{naam:Victoria; geb:23/10/1978; fam:\van Asch\vvan Wijck; tel:24.66.16.9.57;
  fax:24.64.19.5.42; pcd:6601cw; str:Tunnel\weg; hnr:4; wpl:\Wijchen}%
{naam:Hans; geb:1/10; fam:\vdn Bijlaard; tel:26.41.26.20.5;
  pcd:6824jn; str:\kln{}{L\lng{aa}n v\lng{an}}{ }Klarenbeek; hnr:105; wpl:\Arnhem}%
{{naam:Ari\"ette; geb:1/9/1984}{naam:Remelie; geb:11/7/1986};
  fam:\van Elden; tel:26.36.44.34.2;
  pcd:6862zh; str:\kln{}{Pr\lng{ins}}{ }Bernhard\weg; hnr:18; wpl:\Oosterbeek}%
{naam:Xander; fam:\van Es; tel:26.40.13.85.4;
  pcd:6843am; str:Haarlem\weg; hnr:20; wpl:\Arnhem}%
```

Table 9: A few lines of the input file for the lists in table 6, 10 and 12 (data changed for privacy reasons).

Top level TEX code:

```
\begin{phonelist}{AIO \today}{%
  size=A9,eyepos=right,stack,columns=2,outline,%
  flush=left,fontsize=5pt,stretch=1.07,%
  textshape=\sffamily,numbershape=\nrfamily,%
  telarea=26}%
  \begin{fristisort}{naam}
    \input ledenlst.dat
  \end{fristisort}%
  \fristifoot[r]
\end{phonelist}
```

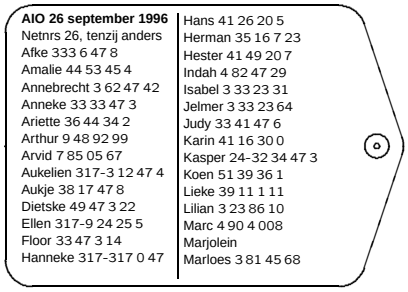


Table 10: Example of a phone list which can be carried on a key fob, based on the input of which some lines are presented in table 9 (only one side shown). We produce this list in a laminated version, with an eye in it. The font size used is five points, but choosing the fonts sensibly (e.g. cmssq for numbers) and using a decent printer bore its fruits: the conductor of the orchestra, who has to use all kinds of glasses to see properly, exclaimed: “even I can read it!”. This phone list also demonstrates the use of the fristisort environment, which sorts the data by the field “naam” (name).

Top level TEX code:

```
\begin{phonelist}{SGA 1A 1995/1996}{%
  width=30mm,height=48.5mm,seal=3mm,%
  eyepos=top,outline,flush=centre,%
  fontsize=5pt,stretch=1.05,%
  textshape=\sffamily,numbershape=\nrfamily}%
  \addvspace{.5\baselineskip}%
  \begin{fristisort}{naam}
    \input sga1a.dat
  \end{fristisort}%
  \fristifoot[c]
\end{phonelist}
```

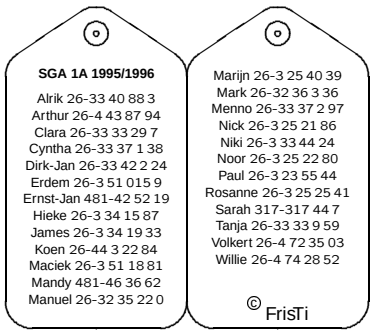


Table 11: Another example of a key fob size phone list.

Top level TEX code:

```
\begin{cakelist}{AIO \today}{%
  size=A9,eyepos=right,stack,columns=2,outline,%
  fontsize=5pt,stretch=1.14,%
  textshape=\sffamily,numbershape=\nrfamily}%
  \input ledenlst.dat
\end{cakelist}
```

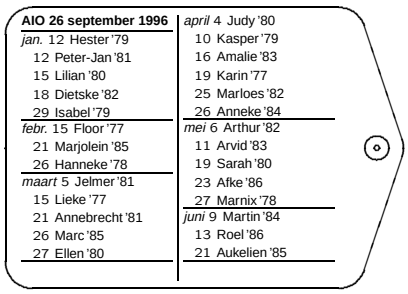


Table 12: Example of a key fob size birthday calendar, based on the input of which some lines are presented in table 9 (only one side shown)

PMGRAPH.STY: some useful macros which extends the LaTeX picture environment

A.S.Berdnikov and O.A.Grineva

berd@ianin.spb.su, olga@ianin.spb.su

The original TeX/LaTeX possibilities to create pictures are relatively poor, and there are many extensions (`epic/eepic`, `pictex`, `drawtex`, `xypic`, `mfpic`, etc.) which were created to extend its possibilities to a higher level. The macro PMGRAPH.STY (*poor-man-graphics*) which are described here are not so general as the ones cited above. They manupulate with the pseudo-graphical fonts which are used by generic LaTeX without additional extensions — mainly because the variations of PiCTeX, METAFONT and new graphical font themes are already realized by other authors and on sufficiently higher level. To some extend the purpose of our work was to see how far it is possible to move in the development of new useful graphical primitives for LaTeX *without* the investment of the external graphical tools.

The style file PMGRAPH.STY includes the following features:

- the vectors with a set of slopes which is as general as the line slopes implemented in LaTeX;
- the vectors with an arrow at the beginning, at the middle or at the end of vector with various orientations of the arrow;
- the circles and circular arcs with nearly arbitrary diameter using magnified `circle` and `circlew` LaTeX fonts;
- the 1/4 circular arcs correctly positioned at the centrum or at the corner;
- extended set of frames which include various corner style and the optional multiple frame shadows with a variety of styles;
- tools which enable the user to extend the variety of frame styles and the shadow styles as far as his/her fantasy allows it;
- automatic calculation of the picture size in terms of the current text width — including the `picture` inserted inside list environments.

Even not very complicated, these macros appears to be useful in our work, and it seems that they can be useful for other TeX-users too.

Vectors

The number of angles for inclined lines which can be used in LaTeX is limited to great extend, but the number of angles for *vectors* is limited even more. The variety of vectors can be extended if instead of the *strictly* inclined arrows at the end of the inclined line the arrow with the *approximate* inclination is added. Corresponding changes are incorporated in PMGRAPH where the relation between strict inclinations and approximate inclinations are shown in Table 1. The corrections require the modifications of the internal LaTeX commands `\@svector`, `\@getlarrow`, `\@getrarrow` and the command `\vector` itself. As a result the command `\vector` starts to draw the vectors for all inclinations valid for LaTeX lines as it is shown on Fig. 1. The vectors are not so ideal as it is required by TeX standards, but the results are acceptable for all inclinations except (6, 1).

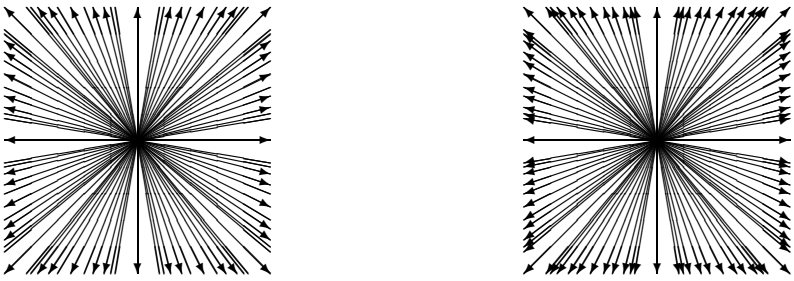


Figure 1: LaTeX and PMGRAPH vectors

(1, 1)	(1, 1)	(4, 1)	(4, 1)	(5, 3)	(3, 2)
(2, 1)	(2, 1)	(4, 3)	(4, 3)	(5, 4)	(4, 3)
(3, 1)	(3, 1)	(5, 1)	(4, 1)	(6, 1)	(4, 1)
(3, 2)	(3, 2)	(5, 2)	(3, 1)	(6, 5)	(4, 3)

Table 1: Relation between line slopes and approximate vector slopes

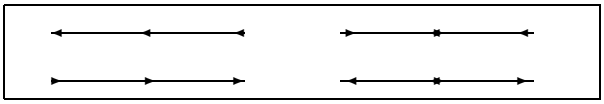


Figure 2: Multi-arrow vectors

LaTeX allows to put an arrow just at the end of the vector. The command `\Vector` enables to put along the vector *arbitrary* arrows with different orientation (see Fig. 2). The predefined arrow styles assign a letter to each position and orientation of the arrow along the `Vector`.

The arrows shown on Fig. 2 are drawn by the commands

```
\begin{picture}(300,40)
\put(20,5){\Vector[bme](1,0){100}}
\put(20,30){\Vector[BME](1,0){100}}
\put(170,5){\Vector[xmMZ](1,0){100}}
\put(170,30){\Vector[XmMz](1,0){100}}
. . . . . .
```

Letter `e` corresponds to normally oriented arrow at the end of vector, `E` — to reverse oriented arrow, `b` and `B` — to (normally and reverse oriented) arrows at the beginning of the vector, `m` and `M` — to the arrows at the middle, etc. The list of letters as the optional parameter produces the set of arrows along the `Vector`. It is possible to create user-defined styles of arrows using the commands `\VectorStyle` and `\VectorShiftStyle` (where the parameters in square brackets are *obligatory*, not *optional*):

`\VectorStyle[`*`style-char`*`]{`*`shift-char`*`}{`*`position`*`}{`*`orientation`*`}`

- *style-char* is the character which is assigned to vector style;
- *shift-char* is the character which defines the relative shift of the arrow with respect to *position* — see command `\VectorShiftStyle` below;
- *position* is the real value which defines the relative position of the arrow along the vector (0.0 means starting point of the vector, 1.0 means end point of the vector) which usually is in a range 0..1 but can be greater 1 or less 0 as well;
- *orientation* is the character which defines the orientation of the arrow with respect to the standard direction of the vecrtor: `b` means *backward* direction, `f` (or any other character) means forward direction.

`\VectorShiftStyle[`*`style-char`*`]{`*`shift`*`}`

- *style-char* is the character which is assigned to vector-shift-style;
- *shift* is the relative shift in `pt` of the arrow along the arrow direction with respect to the positioning point (it is necessary to note that the length of the arrow body in LaTeX is equal to `4pt`).

Examples:

- standard *vector-shift-styles*:
  - `\VectorShiftStyle[e]{0pt}` — style '`e`' means that the end of the arrow is positioned strictly at the point, specified by the parameter *position*;
  - `\VectorShiftStyle[b]{4pt}` — style '`b`' means that the backside of the arrow is positioned at the point, specified by the parameter *position*;
  - `\VectorShiftStyle[m]{3pt}` — style '`m`' means that the middle of the arrow body is positioned at the point, specified by the parameter *position*;
  - `\VectorShiftStyle[E]{-2pt}` — style '`E`' means that the end of the arrow is positioned a little bit before (i.e., by `2pt`) the point, specified by the parameter *position*;
  - `\VectorShiftStyle[B]{6pt}` — style '`B`' means that the backside of the arrow is positioned a little bit after (i.e., by `2pt`) the point, specified by the parameter *position*.

- standard *vector-styles*:
  - `\VectorStyle[e]{e}{1.0}{f}` — style '`e`' means that the end of the arrow is positioned at the end of the vector, and its orientation is along the vector orientation;
  - `\VectorStyle[E]{b}{1.0}{b}` — style '`E`' means that the backside of the arrow is positioned at the end of the vector, and its orientation is rotated by 180° with respect to the vector orientation;
  - `\VectorStyle[b]{b}{0.0}{f}` — style '`b`' means that the backside of the arrow is positioned at the beginning of the vector, and its orientation is along the vector orientation;
  - `\VectorStyle[B]{e}{0.0}{b}` — style '`B`' means that the end of the arrow is positioned at

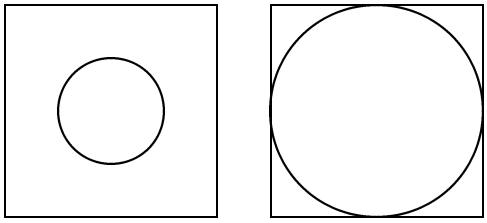


Figure 3: Magnified circles

the beginning of the vector, and its orientation is rotated by 180° with respect to the vector orientation;

`\VectorStyle[m]{m}{0.0}{f}` — style 'm' means that the middle of the arrow body is positioned at the middle of the vector, and its orientation is along the vector orientation;

`\VectorStyle[M]{m}{0.0}{b}` — style 'M' means that the middle of the arrow body is positioned at the middle of the vector, and its orientation is rotated by 180° with respect to the vector orientation;

`\VectorStyle[x]{E}{1.0}{f}` — style 'x' means that the end of the arrow is positioned a little bit before the end of the vector, and its orientation is along the vector orientation;

`\VectorStyle[X]{B}{1.0}{b}` — style 'X' means that the backside of the arrow is positioned a little bit before the end of the vector, and its orientation is rotated by 180° with respect to the vector orientation;

`\VectorStyle[z]{B}{0.0}{f}` — style 'z' means that the backside of the arrow is positioned a little bit after the beginning of the vector, and its orientation is along the vector orientation;

`\VectorStyle[Z]{E}{0.0}{b}` — style 'Z' means that the end of the arrow body is positioned a little bit after the beginning of the vector, and its orientation is rotated by 180° with respect to the vector orientation.

Circles

The range of the diameters for circles and disks (black circular blobs) available in LaTeX is very restricted. It can be enlarged by using the magnified pseudo-graphical LaTeX fonts if the User does not have something better at his/her disposal like `curves.sty`, PiCTeX or MFPiC. The disadvantage of this method is that the width of the lines is magnified too which is inconsistent with the rigorous TeX accuracy requirements, but for *poor man graphics* these circles can be satisfactory.

The scaling of circular fonts is performed by the commands

```
\scaledcircle{factor}
\magcircle{magstep}
```

which are indentical to TeX commands

```
\font ...  scaled factor
\font ...  scaled \magstep magstep
```

The valid *magstep* values are `0`, `h`, `1`, `2`, `3`, `4`, `5`. The values *factor*=1000 and *magstep*=0 correspond to *one-to-one* magnification. The circle magnification like other TeX commands returns to its previous value outside the group inside which it was changed.

To calculate properly the circle character code after the magnification it was necessary to redefine some internal LaTeX commands like `\@getcirc` and `\@circ`. To reflect in magnified fonts the changes of the line thickness, the commands `\thinlines` and `\thicklines` are corrected also.

The example on Fig. 3 is produced by

```
\setlength{\unitlength}{1pt}
\begin{picture}(200,100)(-100,-50)
\put(-50,0){\thicklines\circle{80}}
\put(-50,0){\squareframe{40}}
\magcircle{4}
\put(50,0){\thinlines\circle{80}}
\put(50,0){\squareframe{40}}
\end{picture}
```

where `\squareframe` is the user-defined command which draws the square with the specified side and the centrum at (0,0). It shows how the usage of the magnified circles enables to overcome the upper limit 40pt of the diameter of the LaTeX cirles. It is necessary to note that the thickness of the `\thinline` circles after magnification with `\magcircle{4}` corresponds approximately to the thickness of the ordinary `\thickline` circles (`\magstep4` $\approx$ 2000).

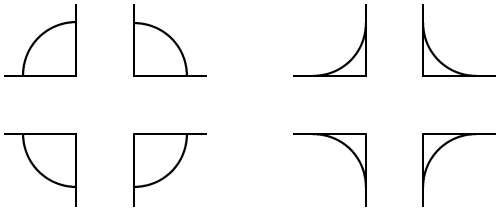


Figure 4: 90° circular segments

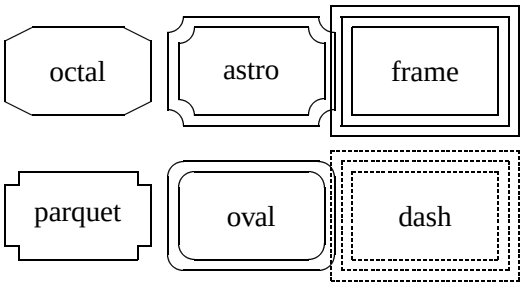


Figure 5: Examples of frame styles

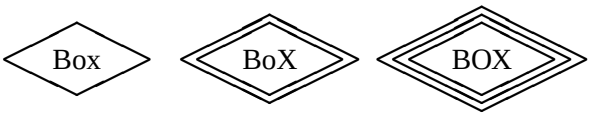


Figure 6: Romb-style frames

Additional macro enable to draw 90° quaters of the circles explicitly without tricky refinement of the parameters of the command `\oval`:

`\trcircle{`*diam*`}` ⟶ `\oval[tr]...`
`\brcircle{`*diam*`}` ⟶ `\oval[br]...`
`\tlcircle{`*diam*`}` ⟶ `\oval[tl]...`
`\blcircle{`*diam*`}` ⟶ `\oval[bl]...`

The centrum of the circular arc is positioned strictly at the point which is the argument of the corresponding `\put`. The commands `\TRcircle`, `\BRcircle`, `\TLcircle`, `\BLcircle` draw the 90° circular quaters with the reference point positioned at the corner instead of the centrum. Similarly, the commands

`\tlsector`, `\TLsector`, `\blsector`, `\BLsector`, ...

draw circular segments together with horizontal and vertical radii. The proper positioning of the circular segments requires special precausions since it is necessary to take into account the line thickness and the specific alignment of the circular elements inside the character boxes.

The example on Fig 4 shows the usage of these commands:

```
\begin{picture}(200,60)(-100,-30)
\put(-60,10){\thicklines\tlcircle{50}}
\put(-60,10){\circle*{1}}
\put(-60,10){\line(-1,0){25}}
\put(-60,10){\line(0,1){25}}
\put(40,10){\thicklines\BRcircle{50}}
\put(40,10){\circle*{1}}
\put(40,10){\line(-1,0){25}}
\put(40,10){\line(0,1){25}}
... ... ...
```

The actual diameter of the circular segment is adjusted like it is done with the circles. The commands `\scaledcircle` and `\magcircle` affect the thickness and the diameter of these circular segments also.

Frames

The set of frames which are available in LaTeX is enhanced in PMGRAPH — except solid and dashed rectangular frames it is possible to draw double and tripple frames in a variety of styles (Fig. 5). The commands `\frameBox`, `\ovalBox`, `\octalBox`, `\astroBox`, `\parquetBox` have the same structure as the command `\framebox`, but they draw the corresponding fancy frames:

```
\put(0,0){\ovalBox(100,50){oval}}
\put(70,0){\astroBox(100,50){astro}}
. . . . . .
```

The ordinary solid frame is drawn by `\frameBox`, the double and triple frames are drawn by `\frameBoX` and `\frameBOX`, respectively. Similar commands exist for double and triple fancy frames. The User can prepare the personal macro commands to draw frame corners and extend the variety of fancy frames up to the limit of his/her fantasy.

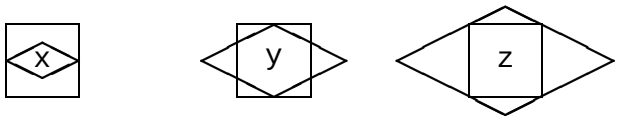


Figure 7: Alignment of romb boxes

More exotic variant of a frame can be created using the commands \rombBox, \rombBoX or \rombBOX as it is shown on Fig. 6. The style (i.e., inclination of the romb sides) and the distance between multiple rombs are set by the command \rombboxstyle

\rombboxstyle(Δx,Δy,*len*) — defines the inclination for the romb boxes and for the corners of the tt octal frames and shadows. Parameters Δx, Δy specifies the inclination, and the parameter *len* — the length of the inclined corners (for octal frames and shadows only) in a style similar to the command \line(Δx,Δy){*len*}.

with the default settings as

```
\rombboxstyle(2,1,2pt)
```

The alignment of the romb around the box specified for these commands can be varied using additional optional parameter(see Fig. 7. The full format of the rombbox commands is:

\rombBox[*char*](ΔX,ΔY){*text*}

where *char* -is one-character parameter which defines the alignment of the romb frame with respect to rectangle (ΔX, ΔY): x (default value) means that the x-axis coinsides with the x-axis of the rectangle, y means that the y-axis coinsides with the y-axis of the rectangle, z means that the corners of the rectangle are at the sides of the romb frame.

Each rectangular box has the optional parameter which enable to specify the "shadows" around this box. Each shadow style has a special letter, and the list of letters as the optional parameter draws a list of shadows. The standard shodow types are shown on Fig. 9. It is possible to draw several shadows of different types and around arbitrary corner of the frame as it is shown on Fig. 10:

```
\unitlength=10pt
\begin{picture}(20,15)
\shadowcorner{B}
\put(0,0){\frameBoX[oPR...](10,5){...}}
\end{picture}
```

The parameter of the shadows — thickness, corner size, additional shift, etc., — can be varied by the following User commands:

\framesep{*dist*} — set the distance between double and triple frames. It can be negative as well as positive. Default value: 2pt}.

\shadowsep{*dist*} — set the gap distance between the frame and the shadow or between multiple shadows. Default value: 1pt}.

\shadowsize{*dist*} — set the depth of the shadow. Default value: 5pt}.

\shadowshrink{*factor*} — set the contraction factor for the subsequent shadows. Default value: 1}.

\shadowcorner{*char*} — set the corner for the shadows. Valid values: A, B, C, D. Default value: \shadowcorner{A}.

\RoundCorner{*radius*} — set the *radius* for the circular arcs at the corners of oval frames and shadows with rounded corners. Default value: 5pt{5pt}.

\DiskCorner{*diam*} — set the *diameter* for the bulbs at the corners of black shadows with rounded corners. Default value: 5pt{5pt}.

\LineCorner{*len*} — set the *length* for the inclined corners of octal frames and shadows. Default value: 10pt{10pt}.

\RectCorner{*size*} — set the *size* for the parquette corners of octal frames and shadows. Default value: 5pt{5pt}.

Rombs

Special command enable to draw rombs (see Fig. 8):

\put(x)(y) {\romb[*pos*](Δx,Δy){*len*}}

where:

(x, y) — position of the romb inside picture;
pos — one-character option which shows the alignment of romb with respect to (x, y): r means right corner, l means left corner; c means center (default);
(Δx, Δy) and *len* are the parameters which define the inclination and the length of the romb side (similarly to \line).

Automatically scaled pictures

The idea of the macros which are responsible for these functions is to calculate the \unitlength value in terms of the *relative fraction* of the page width instead of explicit specifying its value in points, centimeters, inches, etc.

The command \pictureunit[*percent*]{*x-size*} selects the value of the variable \unitlength so that the picture which is *x-size* units in width occupies *percent* width of the paper. The environment Picture combines the automatic calculation of the \unitlength with the \begin{picture} – \end{picture}.

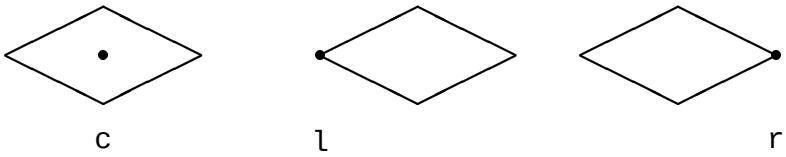


Figure 8: Alignment of rombes

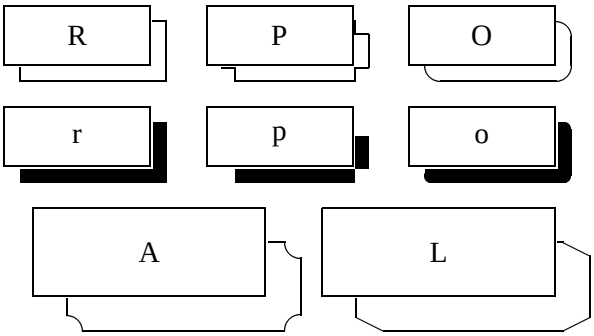


Figure 9: Examples of shadows

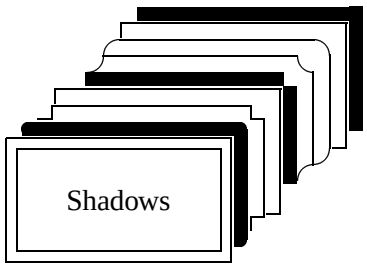


Figure 10: Multiple shadows

By default *percent*=100 is used which corresponds to 90% of the paper width. The default *percent* value can be redefined by the command

`\def\defaultpercent{`*percent*`}`.

Examples:

```
\pictureunit[75]{120}
\begin{picture}(120,80)
...
\end{picture}

\begin{Picture}[75](120,80)
...
\end{Picture}
```

These macros are inspired by `fullpict.sty` by Bruce Shawyer. Carefull examination of the file `fullpict.sty` shows some bugs/warnings which require correction:

- each automatic scaling of `\unitlength` allocates a new counter;
- automatic scaling uses `\textwidth` as the reference width which results to improper functioning inside list and `minipage` environments;
- the environments `fullpicture`, `halfpicture` and `scalepicture` are centered internally with `\begin{center}` — `\end{center}` which prevents the proper positioning of the picture in most cases.

The PMGRAPH.STY macros calculates the dimension \unitlength using the value \hsize, and as a result it works corectly also for twocolumn mode, inside the *list* environments itemize, enumerate, etc. (for example, all the figures in this paper are drawn using the environment Picture). The automatic centering and repeatedly allocation of the registers are corrected as well.

Acknowledgements

The authors are grateful to Dr. Kees van der Laan for the possibility to present the results of our research at the EUROTEX'95 (Arnhem, Netherlands).

One of the authors (A.S.Berdnikov) would like to express his warmest thanks to Dr. A. Compagner from the Delft University of Technology who spent a lot of his time and efforts trying to transform two naive students from Russia (namely, him and his co-worker Sergey Turtia) into serious scientists.

This research was partially supported by a grant from the Dutch Organization for Scientific Research (NWO grant No 07–30–007).

Using EPS Graphics in LaTeX2ε Documents

Keith Reckdahl *

reckdahl@leland.stanford.edu

Version 1.8c
July 28, 1996

Abstract

This document explains how to use Encapsulated PostScript (EPS) files in LaTeX2ε documents. The graphics and graphicx packages provide commands which insert, scale, and rotate EPS graphics. The following EPS-inclusion topics are covered

- Compressed EPS files and non-EPS graphic formats (TIFF, GIF, JPEG, PICT, etc.) can also be inserted when `dvips` is used. Since neither LaTeX nor `dvips` has any built-in decompression or graphics-conversion capabilities, that software must be provided by the user.
- Since many applications which produce EPS files support only ASCII text, the PSfrag system allows text in EPS files to be replaced with LaTeX symbols or mathematical expressions.
- When an EPS graphic is inserted multiple times, the final PostScript includes multiple copies of the graphics, making the file large. A smaller final PostScript file results from defining a PostScript command for the graphics. An example of inserting an EPS graphic in the page header with the fancyhdr package is provided.

Various commands are often used in conjunction with EPS graphics. The following topics are covered in this document

- the insertion of graphics in `figure` environments allows the graphics to float for better formatting and allows graphics to be referenced,
- the lscape and rotating packages provide methods for producing figures with landscape orientation in a portrait document,
- methods for arranging side-by-side graphics. The graphics can be placed in a single figure, in multiple figures contained in a single float, or in subfigures.
- boxed figures can be created with `\fbox` or the commands from the fancybox package.
- the figure captions can be placed next to the figure or table, instead of the conventional above or below placement,
- the caption2 package adds flexibility to the caption formatting, allowing users to modify the style, width, and font of captions.

Contents

I Background Information **73**

1 Introduction **73**

2 LaTeX Terminology **74**

3 The EPS BoundingBox **74**
3.1 Converting PS files to EPS 75

4 Graphics in DVI Files **75**

II Graphics Inclusion Commands **75**

5 The Commands in the graphicx Package **76**
5.1 The includegraphics Command 76
5.2 The scalebox Command 78
5.3 The resizebox Commands 78
5.4 The rotatebox Command 78

6 The graphics Version of includegraphics **79**

III Importing EPS Graphics **80**

11 The figure Environment **92**
11.1 Caption Vertical Spacing . . . 93
11.2 Figure Placement Options . . . 93

12 Landscape Figures **94**
12.1 Landscape Environment . . . 94
12.2 Sidewaysfigure Environment . . . 94
12.3 Rotcaption Command . . . 95

13 Side-by-Side Graphics **95**
13.1 Side-by-Side Graphics in a Single Figure . . . 95
13.2 Side-by-Side Figures . . . 98
13.3 Side-by-Side Subfigures . . . 100

14 Minipage Placement Option Details **102**
14.1 Aligning the Bottoms of Minipages . . . 103
14.2 Aligning the Tops of Minipages . . . 103

15 Boxed Figures **104**
15.1 Box Around Graphic . . . 104
15.2 Box Around Figure and Caption . . . 104
15.3 Customizing fbox Parameters . . . 105
15.4 The Fancybox Package . . . 105

16 Customizing Captions **107**
16.1 Captions Next to Figures . . . 107
16.2 Controlling Caption Width . . . 107
16.3 Caption Package . . . 108

Acknowledgements **114**

References **114**

Part I

Background Information

1 Introduction

Inserting Encapsulated PostScript (EPS) graphics in LaTeX originally required the low-level `\special` command. To make graphic-insertion easier and more portable, two higher-level packages epsf and ps g were written for LaTeX2.09. In epsf, the graphics insertion was done by the `\epsfbox` command, while three other commands controlled graphic scaling. In ps g, the `\psfig` command not only inserted graphics, it also scaled and rotated them. While the `\psfig` syntax was popular, its code was not as robust as `\epsfbox`. The eps g package was created as a hybrid of the two graphics packages, with its `\epsfig` command using the `\psfig` syntax and much of the more-robust `\epsfbox` code. Unfortunately, `\epsfig` still used some of the less-robust `\psfig` code.

The eps g package was updated to LaTeX2ε as a stop-gap measure while the LaTeX3 team addressed the general problem of inserting graphics in LaTeX2ε. The commands in the totally re-written "graphics bundle" are more efficient and more robust than those in other packages.

The graphics bundle contains the "standard" graphics package and the "extended" graphicx package. While both packages contain an `\includegraphics` command which includes graphics, the packages contain *different* versions of `\includegraphics`. The syntax of the graphicx `\includegraphics` is modeled after `\psfig`, while the syntax of the graphics `\includegraphics` is modeled after the `\epsfbox` command. The graphicx `\includegraphics` supports scaling and rotating, but the graphics `\includegraphics` command must be nested inside `\rotatebox` and/or `\scalebox` commands for scaling and/or rotating.

This document uses the graphicx package because its syntax is more convenient than the graphics syntax. Since both packages have the same capabilities, the examples in this document can also be performed with the graphics package, although the resulting syntax may be more cumbersome. The syntax of the graphicx commands is described in section 5. The syntax of the graphics commands is described in section 6. For a more-detailed description of the packages, see David Carlisle's graphics bundle documentation [1].

For backward-compatibility, the graphics bundle also includes the `eps g` package which replaces the original LaTeX2ε `eps g` package. The new `eps g` package defines the `\epsfbox`, `\psfig`, and `\epsfig` commands as wrappers which simply call the `\includegraphics` command.

2 LaTeX Terminology

A *box* is any LaTeX object (characters, graphics, etc.) that is treated as a unit (see [4, page 103]). Each box has a *reference point* on its left side. The box's *baseline* is a horizontal line which passes through the reference point (see Figure 1). When LaTeX forms lines of text, characters are placed left-to-right with their reference points aligned on a horizontal line called the *current baseline*, aligning the characters' baselines with the current baseline. LaTeX follows the same process for typesetting graphics or other objects; the reference point of each object is placed on the current baseline.

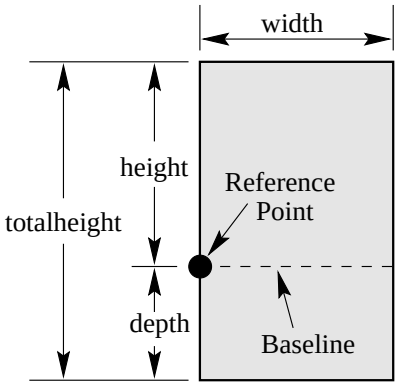


Figure 1: Sample LaTeX Box

The size of each box is described by three lengths: *height*, *depth*, *width*. The *height* is the distance from the reference point to the top of the box. The *depth* is the distance from the reference point to the bottom of the box. The *width* is the width of the box. The *totalheight* is defined as the distance from the bottom of the box to the top of the box, or totalheight=height+depth.

The reference point of a non-rotated EPS graphic is its lower-left corner (see left box in Figure 2), giving it zero depth and making its totalheight equal its height. The middle box in Figure 2 shows a rotated graphic where the height is not equal to the totalheight. The right box in Figure 2 shows a rotated graphic where the height is zero.

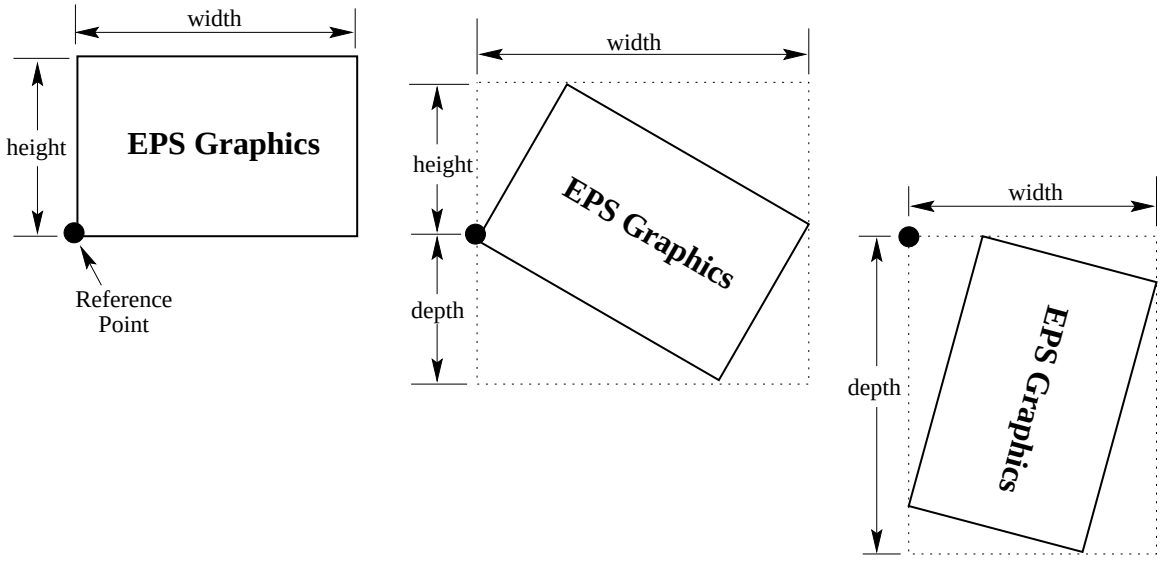


Figure 2: Rotated LaTeX Boxes

3 The EPS BoundingBox

In addition to PostScript graphics language commands which draw the graphics, EPS files contain a BoundingBox line which specifies the natural size of the graphics. By convention, the first line of a PostScript file specifies the type of Post-

Script and is then followed by a series of comments called the *header* or *preamble*. (Like LaTeX, PostScript's comment character is %). One of these comments specifies the BoundingBox. The BoundingBox line contains four integers

1. The x-coordinate of the lower-left corner of the BoundingBox.
2. The y-coordinate of the lower-left corner of the BoundingBox.
3. The x-coordinate of the upper-right corner of the BoundingBox.
4. The y-coordinate of the upper-right corner of the BoundingBox.

For example, the first 5 lines of an EPS file created by gnuplot are

```
%!PS-Adobe-2.0 EPSF-2.0
%%Creator: gnuplot
%%DocumentFonts: Times-Roman
%%BoundingBox: 50 50 410 302
%%EndComments
```

Thus the gnuplot EPS graphic has a lower-left corner with coordinates (50, 50) and an upper-right corner with coordinates (410, 302). The BoundingBox parameters have units of PostScript points which are $1/72$ of an inch, making the above graphic's natural width 5 inches and its natural height 3.5 inches. Note that a PostScript point is is slightly larger than a TeX point which is $1/72.27$ of an inch. In TeX and LaTeX, PostScript points are called "big points" and abbreviated `bp` while TeX points are called "points" and abbreviated `pt`.

3.1 Converting PS files to EPS

While most PostScript files (without BoundingBox information) can be converted to EPS, there are restrictions on the PostScript commands which can be used in EPS files. For example, EPS files cannot include the `setpagedevice`, `letter`, or `a4` PostScript operators. Single-page PostScript files without any such offending commands can be converted to EPS by one of the following methods

1. The best option is to use a utility such as ghostscript's `ps2epsi` which reads the PostScript file, calculates the BoundingBox parameters, and creates an EPS file (complete with a BoundingBox) which contains the PostScript graphics. Unfortunately, ghostscript is a large package which is not trivial to install.
2. Alternatively, the BoundingBox parameters can be calculated and entered in the `bb` option of `\includegraphics` or a text editor can be used to insert them directly in the PostScript file's BoundingBox line. There are several ways to calculate the BoundingBox
   1. The `bbfig` script uses a PostScript printer to calculate the BoundingBox. `bbfig` adds some PostScript commands to the beginning of the PostScript file and sends it to the printer. At the printer, the added PostScript commands calculate the BoundingBox of the original PostScript file, printing the BoundingBox coordinates superimposed on the PostScript graphic.
   2. Use ghostview to display the PostScript graphic. As you move the ghostview pointer around the graphic, ghostview displays the pointer's coordinates (with respect the the lower-left corner of the page). To determine the BoundingBox parameters, record the pointer coordinates at the lower-left corner of the graphic and the upper-right corner of the graphic.
   3. Print out a copy of the PostScript graphics and measure the horizontal and vertical distances (in inches) from the lower-left corner of the paper to the lower-left corner of the graphics. Multiply these measurements by 72 to get the coordinates of the BoundingBox's lower-left corner. Likewise, measure the distances from the lower-left corner of the paper to the upper-right corner of the graphics to get the coordinates of the BoundingBox's upper-right corner.

4 Graphics in DVI Files

When LaTeX documents are compiled, the graphics-inclusion commands do not insert the EPS graphics file into the DVI file. Rather, they do two things

1. They reserve the proper amount of space for the graphic in the LaTeX document.
2. They place a file-specification command in the DVI file which specifies the name of the EPS file.

When a DVI-to-PS converter (such as `dvips`) converts the DVI file to PostScript, the file-specification command causes the converter to insert the EPS graphics into the PostScript file. Therefore,

- the EPS graphics do not appear in most DVI-viewers. To help the user with placement of the graphics, most DVI viewers display the BoundingBox in which the graphics will be inserted.
- the EPS files must be present when the DVI file is converted to PS. Thus the EPS files must accompany DVI files whenever they are moved.

Part II

Graphics Inclusion Commands

5 The Commands in the graphicx Package

The best reference for the graphics and graphicx packages is the graphics guide [1]. The coverage of the graphicx package in the standard LaTeX references is sporadic: [6] covers both the graphics and graphicx packages, [4] only covers the graphics package and [5] describes neither.

The graphicx package has five main commands

```
\includegraphics[options]{filename}
\rotatebox[options]{angle}{argument}
\scalebox{h-scale}[v-scale]{argument}
\resizebox{width}{height}{argument}
\resizebox*{width}{totalheight}{argument}
```

5.1 The includegraphics Command

Syntax: `\includegraphics[options]{filename}`

where the options are listed in Tables 1, 2, and 3. Since `\includegraphics` does not end the current paragraph, it can place EPS graphics within text such as ⊗ or ◔. The commands

```
\documentclass{article}
\usepackage{graphicx}
\begin{document}
   \includegraphics{file.eps}
\end{document}
```

include the graphics from `file.eps` at its natural size.

Specifying Width

The command

```
\includegraphics[width=3in]{file.eps}
```

includes the graphics from `file.eps` scaled such that its width is 3 inches. The command

```
\includegraphics[width=\textwidth]{box.eps}
```

scales the included graphic such that it is as wide as the text. The command

```
\includegraphics[width=0.80\textwidth]{box.eps}
```

makes the included graphic 80% as wide as the text. The following commands make the graphic width 2 inches less than the width of text

```
\newlength{\epswidth}
\setlength{\epswidth}{\textwidth}
\addtolength{\epswidth}{-2.0in}
\includegraphics[width=\epswidth]{box.eps}
```

If the calc package is available, this is shortened to

```
\newlength{\epswidth}
\setlength{\epswidth}{\textwidth -2.0in}
\includegraphics[width=\epswidth]{box.eps}
```

The `\newlength` command only needs to be issued once. Subsequent graphics can be scaled without re-issuing the `\newlength` command. The length name `\epswidth` is not special. Any other name (which isn't already used by LaTeX) could have been used. The calc package with the 12/95 graphicx package shortens this further to

```
\includegraphics[width=\textwidth-2.0in]{box.eps}
```

Specifying Height

Users must be careful when using the `height` option, as they often mean the overall height which is set by the `totalheight` option (see Figure 1). When the object has zero depth, the `totalheight` is the same as the `height` and specifying `height` works fine. When the object has a non-zero depth, specifying `height` instead of `totalheight` causes either an incorrectly-sized graphic or a divide-by-zero error.

Graphic Justification

The placement of the graphic is controlled by the current text justification. To center the graphic, put it inside a center environment

`height`	The height of the graphics (in any of the accepted TEX units).
`totalheight`	The totalheight of the graphics, in any of the accepted TEX units. *(Added 6/95)*
`width`	The width of the graphics, in any of the accepted TEX units.
`scale`	Scale factor for the graphic. Specifying `scale=2` makes the graphic twice as large as its natural size.
`angle`	Specifies the angle of rotation, in degrees, with a counter-clockwise (anti-clockwise) rotation being positive.
`origin`	The `origin` command specifies what point to use for the rotation origin. By default, the object is rotated about its reference point. *(Added 12/95)* The possible origin points are the same as those for the `\rotatebox` command in section 5.4. For example, `origin=c` rotates the graphic about its center.
`bb`	Specifies BoundingBox parameters. For example `bb=10 20 100 200` specifies that the BoundingBox has its lower-left corner at (10,20) and its upper-right corner at (100,200). Since `\includegraphics` automatically reads the BoundingBox parameters from the EPS file, the `bb` option is usually not specified. It is useful if the BoundingBox parameters in the EPS file are missing or are incorrect. The BoundingBox can be specified with the `natheight` and `natwidth` options instead of the `bb` options. `natheight=h, natwidth=w` is equivalent to `bb=0 0 h w`. For backward compatibility, the BoundingBox coordinates can also be individually specified with `bbllx`, `bblly`, `bburx`, `bbury` options.

Table 1: `includegraphics` Options

`viewport`	Specify what portion of the graphic to view. Like a BoundingBox, the area is specified by four numbers which are the coordinates of the lower-left corner and upper-right corner. The coordinates are relative to lower-left corner of the BoundingBox. *(Added 6/95)* For example, `viewport=0 0 72 72` displays the 1-inch square from the lower left of the graphic. Note that some early graphicx versions may have a broken `viewport` option in which `viewport=a b c d` produces an upper-right corner of (a+c,b+d) instead of (c,d).
`trim`	An alternate method for specifying what portion of the graphic to view. The four numbers specify the amount to remove from the left, bottom, right, and top side, respectively. Positive numbers trim from a side, negative numbers add to a side. *(Added 6/95)* For example, `trim=1 2 3 4` trims the graphic by 1 `bp` on the left, 2 `bp` on the bottom, 3 `bp` on the right, 4 `bp` on the top.

Table 2: `includegraphics` Cropping Options

```
\begin{center}
   \includegraphics[width=2in]{box.eps}
\end{center}
```

If the `\includegraphics` command is inside an environment (such as `minipage` or `figure`), the `\centering` declaration centers the remaining output of the environment. For example

```
\begin{figure}
   \centering
```

`clip`	When `clip` is specified, any graphics outside of the viewing area are clipped and do not appear.
`keepaspectratio`	When `keepaspectratio` is not specified, specifying both the `width` and either `height` or `totalheight` causes the graphic to be scaled anamorphically to fit both the specified height and width. When `keepaspectratio` is specified, specifying both the `width` and either `height` or `totalheight` makes the graphic as large as possible such that its aspect ratio remains the same and the graphic does exceeds neither the specified height nor width. *(Added 9/95)*
`draft`	When `draft` is specified, the graphic's BoundingBox and filename are displayed in place of the graphic, making it faster to display and print the document. The `draft` package option `\usepackage[draft]{graphicx}` causes all the graphics in a document to be inserted in draft mode.

Table 3: `includegraphics` Boolean Options

```
   \includegraphics[width=2in]{box.eps}
\end{figure}
```

is similar to

```
\begin{figure}
\begin{center}
   \includegraphics[width=2in]{box.eps}
\end{center}
\end{figure}
```

The difference between these examples is that the `center` environment produces extra vertical space above and below the environment, while `\centering` produces no extra space.

5.2 The scalebox Command

Syntax: `\scalebox{h-scale}[v-scale]{argument}`

The `\scalebox` command scales an object, making its width be its original width multiplied by `h-scale`. The object can be any LaTeX object: letter, paragraph, EPS graphic, etc. The object's height is its original height multiplied by `v-scale`. Negative values reflect the object. If `v-scale` is omitted, it defaults to h-scale, which keeps the aspect ratio constant.

5.3 The resizebox Commands

Syntax: `\resizebox{width}{height}{argument}`
`\resizebox*{width}{totalheight}{argument}`

The `\resizebox` command resizes an object to a specified size. The object can be any LaTeX object: letter, paragraph, EPS graphic, etc. Specifying `!` as either height or width makes that length be such that the aspect ratio remains constant. The standard LaTeX2ε arguments `\height`, `\width`, `\totalheight`, `\depth` can be used to refer to the original size of `argument`. So `\resizebox{2in}{\height}{argument}` makes `argument` keep its same height but have a width of 2 inches.

The `\resizebox*` command is identical to `\resizebox`, except the second argument specifies the `totalheight` of the object.

5.4 The rotatebox Command

Syntax: `\rotatebox[options]{angle}{argument}`

The `\rotatebox` command rotates an object by an angle given in degrees, with a counter-clockwise rotation being positive. The object can be any LaTeX object: letter, paragraph, EPS graphic, etc. By default, the object is rotated about its reference point. The options allow the user to specify the point of rotation

1. Specifying the `[x=xdim,y=ydim]`, the object is rotated about the point whose coordinates relative to the reference point are (`xdim,ydim`).

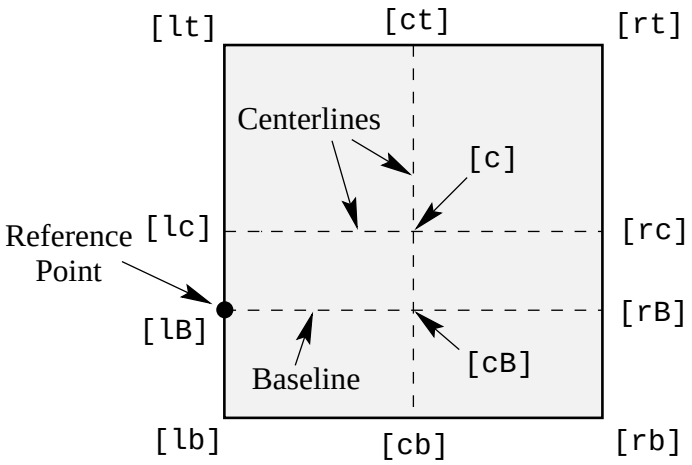


Figure 3: Available Origin Points

2. The `origin` option specifies one of 12 special points shown in in Figure 3.
   The horizontal position of the `origin` points is specified by one of three letters: `lcr` (which stand for left, center, right, respectively), while the vertical position is specified by one of four letters: `t,c,B,b` (which stand for top, center, Baseline, bottom, respectively). For example
   `[rb]` specifies the bottom-right corner
   `[lt]` specifies the top-left corner
   `[cB]` specifies the center of the graphic's Baseline
   `[lc]` specifies the midpoint of the left side
   `[ct]` specifies the midpoint of the top side

   Note that
   - The order of the letters is not important, making `[br]` equivalent to `[rb]`.
   - `c` represents either the horizontal center or vertical center depending what letter is used with it.
   - If only one letter is specified, the other is assumed to be `c`, making `[c]` equivalent to `[cc]`, `[l]` equivalent to `[lc]`, `[t]` equivalent to `[tc]`, etc.

6 The graphics Version of includegraphics

This document uses the graphicx package because its syntax is more convenient than the graphics syntax. Since both packages have the same capabilities, the examples in this document can also be performed with the graphics package, although the resulting syntax may be more cumbersome.

The graphics package contains two commands `\includegraphics` and `\includegraphics*` which are identical except that `\includegraphics*` clips (does not show) graphics outside the BoundingBox. The syntax for `\includegraphics` is

```
\includegraphics[llx,lly][urx,ury]{filename}
```

`[llx,lly]` are the *x* and *y* coordinates of the lower-left corner of the image. `[urx,ury]` are the *x* and *y* coordinates of the upper-right corner of the image. If no coordinates are given, the BoundingBox in the file is used. If only one set of coordinates is listed, it is assumed to be `[urx,ury]`, with `[llx,lly]` set to zero. The default units for the coordinates are `bp`, although any valid TeX units can be used.

The graphics package's `\rotatebox`, `\scalebox`, `\resizebox` commands are the same as the corresponding graphicx commands except the graphics version of `\rotatebox` does not allow any of the options which the graphicx version offers (see section 5.4).

The following commands use the graphicx version of `\includegraphics`

```
\documentclass{article}
\usepackage{graphicx}
\begin{document}
   %% include file1.eps with a width of 3 inches
   \includegraphics[width=3in]{file1.eps}
```

```
    %% include file2.eps with a width of 3 inches, then rotate 45 degrees
    \includegraphics[width=3in,angle=45]{file2.eps}

    %% include file3.eps, rotate 45 degrees, and resize to a width of 3 inches
    \includegraphics[angle=45,width=3in]{file3.eps}
\end{document}
```

The same output can be produced by the graphics version of `\includegraphics`

```
\documentclass{article}
\usepackage{graphics}
\begin{document}
   %% include file1.eps with a width of 3 inches
   \resizebox{3in}{!}{\includegraphics{file1.eps}}

   %% include file2.eps with a width of 3 inches, then rotate 45 degrees
   \rotatebox{45}{\resizebox{3in}{!}{\includegraphics{file2.eps}}}

   %% include file3.eps, rotate 45 degrees, and resize to a width of 3 inches
   \resizebox{3in}{!}{\rotatebox{45}{\includegraphics{file3.eps}}}
\end{document}
```

Part III

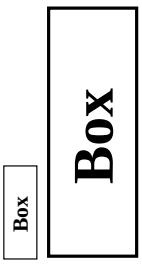
Importing EPS Graphics

7 Rotation and Scaling

Since the `\includegraphics` options are interpreted from left to right, the order in which the angle and size are specified makes a difference. For example

```
\begin{center}
   \includegraphics[angle=90,totalheight=0.5in]{box.eps}
   \includegraphics[totalheight=0.5in,angle=90]{box.eps}
\end{center}
```

produces



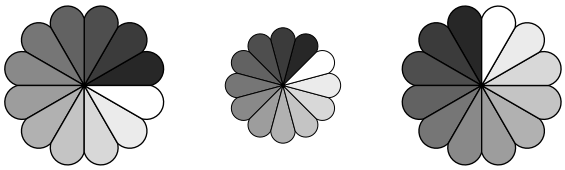
The first box is rotated 90 degrees and then scaled such that its height is a half inch. The second box is scaled such that its height is a half inch and then it is rotated 90 degrees.

7.1 Scaling of Rotated Graphics

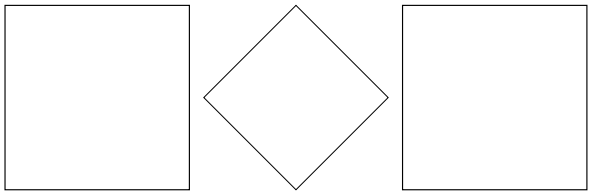
When the height or width of a graphic is specified, the specified size is not the size of the graphic but rather of its BoundingBox. This distinction is especially important in order to understand the scaling of rotated graphics. For example

```
\begin{center}
  \includegraphics[totalheight=1in]{rosette.eps}
  \includegraphics[angle=45,totalheight=1in]{rosette.eps}
  \includegraphics[angle=90,totalheight=1in]{rosette.eps}
\end{center}
```

produces



Although it may seem strange that the graphics have different sizes, it makes sense after viewing the BoundingBoxes



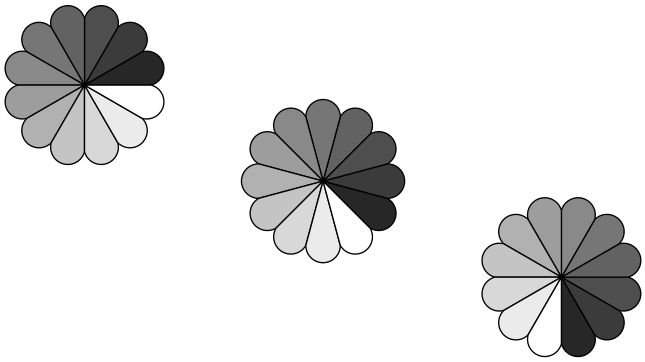
Each graphic is scaled such that its rotated BoundingBox is 1 inch tall.

7.2 Alignment of Rotated Graphics

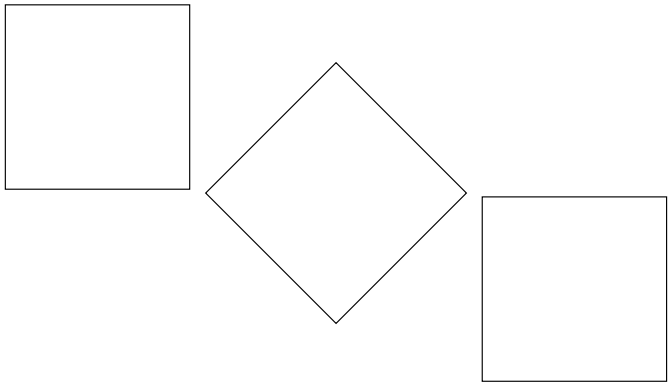
When graphics are rotated, the objects may not align properly. For example

```
\begin{center}
  \includegraphics[totalheight=1in]{rosette.eps}
  \includegraphics[totalheight=1in,angle=-45]{rosette.eps}
  \includegraphics[totalheight=1in,angle=-90]{rosette.eps}
\end{center}
```

produces



Again, this is better illustrated by the BoundingBoxes



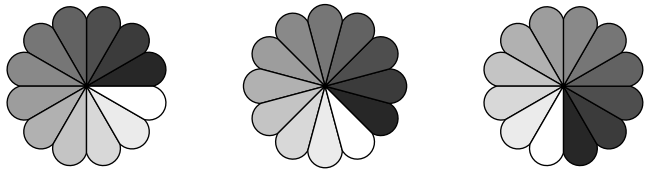
In this case, the objects' reference points (original lower-left corners) are aligned on a horizontal line. If it is desired to instead have the centers aligned, the `minipage` environment can be used

```
\begin{center}
   \begin{minipage}[c]{1.25in}
      \centering
      \includegraphics[totalheight=1in,angle=0]{rosette.eps}
   \end{minipage}
   \begin{minipage}[c]{1.25in}
      \centering
      \includegraphics[totalheight=1in,angle=-45]{rosette.eps}
   \end{minipage}
   \begin{minipage}[c]{1.25in}
      \centering
      \includegraphics[totalheight=1in,angle=-90]{rosette.eps}
   \end{minipage}
\end{center}
```

However, it is easier to use the `\rotatebox` command to rotate the graphic about its center

```
\begin{center}
   \includegraphics[totalheight=1in]{rosette.eps}
   \rotatebox[origin=c]{-45}{\includegraphics[totalheight=1in]{rosette.eps}}
   \rotatebox[origin=c]{-90}{\includegraphics[totalheight=1in]{rosette.eps}}
\end{center}
```

This aligns the centers of the graphics



If the 12/95 version of graphicx is used, the `origin` option can be used in `\includegraphics`

```
\begin{center}
  \includegraphics[totalheight=1in]{rosette.eps}
  \includegraphics[totalheight=1in,origin=c,angle=-45]{rosette.eps}
  \includegraphics[totalheight=1in,origin=c,angle=-90]{rosette.eps}
\end{center}
```

Similarly, the commands

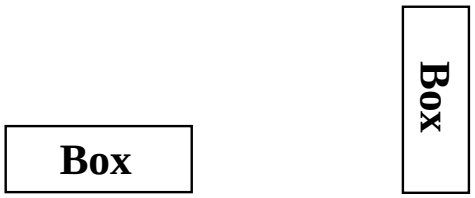
```
\begin{center}
    \includegraphics[width=1in]{box.eps}
    \hspace{1in}
    \includegraphics[width=1in,angle=-90]{box.eps}
\end{center}
```

rotate the right box around its lower-left corner, producing

To align the bottoms of the graphics, use the following commands

```
\begin{center}
    \includegraphics[width=1in]{box.eps}
    \hspace{1in}
    \rotatebox[origin=br]{-90}{\includegraphics[width=1in]{box.eps}}
 \end{center}
```

to rotate the right box around its lower-right corner



If the 12/95 version of graphicx is used, the `origin` option can be used in `\includegraphics`

```
\begin{center}
   \includegraphics[width=1in]{box.eps}
   \hspace{1in}
   \includegraphics[width=1in,origin=br,angle=-90]{box.eps}}
\end{center}
```

8 Compressed and Non-EPS Graphics Files

When using `dvips`, users can specify an operation to be performed on the file before it is inserted. By making this operation a decompression command, compressed graphics files can be used. By making this a graphics-conversion command, non-EPS graphics files can be used. Since `dvips` is currently the only DVI-to-PS converter with this capability, everything in this section requires `dvips`. Users need to pass the `dvips` option to the graphicx package. This can be done by either specifying the `dvips` global option in the `\documentclass` command

```
\documentclass[dvips,11pt]{article}
```

or by specifying the `dvips` option in the `\usepackage` command

```
\usepackage[dvips]{graphicx}
```

Since specifying the `dvips` option in `\documentclass` passes it to all packages, it is generally preferred.

The `\DeclareGraphicsRule` and `\DeclareGraphicsExtensions` commands control how LaTeX deals with files specified in `\includegraphics`. In particular, `\DeclareGraphicsRule` specifies a command which operates on the file. The execution of this command requires that the operating system support pipes. Without piping, the decompression or conversion cannot be done on-the-fly and the user must store all graphics as uncompressed EPS files.

8.1 Compressed EPS Example

The steps for using compressed EPS files are

1. Create an EPS file (`file1.eps` for example)
2. Store the BoundingBox line in another file (`file1.eps.bb`)
3. Compress the EPS file. For example, the Unix command `gzip -9 file1.eps` creates the compressed file `file1.eps.gz`. The `-9` (or `-best`) option specifies maximum compression.
4. Include the proper `\DeclareGraphicsRule` command before the `\includegraphics` command in the LaTeX file. The `\DeclareGraphicsRule` command informs LaTeX how to treat the particular suffix (see section 8.2). For example

```
\documentclass[dvips]{article}
\usepackage{graphicx}
\begin{document}
   \DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{‘gunzip -c #1}
   \begin{figure}
        \centering
        \includegraphics[width=3in]{file1.eps.gz}
        \caption{Compressed EPS Graphic}\label{fig:compressed:eps}
   \end{figure}
\end{document}
```

In this case, the `\DeclareGraphicsRule` command is actually not necessary because it happens to be one of the pre-defined graphics rules. If another compression program or suffixes were used, the `\DeclareGraphicsRule` command would be mandatory. For example, if the BoundingBox file had been stored in `file1.bb`, the corresponding `\DeclareGraphicsRule` would be

```
\DeclareGraphicsRule{.eps.gz}{eps}{.bb}{‘gunzip -c #1}
```

If many compressed EPS files are used, the BoundingBox extraction and EPS file compression may be tedious. This process can be automated with a `perl` script named `epsbb` which was included with the old `eps g` package and is still available from CTAN.

8.2 The DeclareGraphicsRule Command

The `\DeclareGraphicsRule` command specifies how `\includegraphics` treats files depending on their extensions. Multiple `\DeclareGraphicsRule` commands may be issued.

Syntax: `\DeclareGraphicsRule{ext}{type}{sizefile}{command}`

`ext`	The file extension.
`type`	The graphics type for that extension.
`sizefile`	The extension of the file which contains the BoundingBox information for the graphics. If this option is blank `{}`, the size information must be specified by an `\includegraphics` option.
`command`	The command to be applied to the file (often left blank `{}`). The command must be preceded by a single backward quote (not to be confused with the more common forward single quote.)

Table 4: `DeclareGraphicsRule` Arguments

For example, the command

```
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{‘gunzip -c #1}
```

specifies that any file with a `.eps.gz` extension is treated as compressed EPS file, with the the BoundingBox information stored in the file with a `.eps.bb` extension, and the `gunzip -c` command uncompresses the file. (Since LaTeX cannot read BoundingBox information from a compressed file, the BoundingBox line must be stored in an uncompressed file.)

Users generally do not need to use the `\DeclareGraphicsRule` command because the following graphics rules are defined by default in `dvips.def`

```
\DeclareGraphicsRule{.eps}{eps}{.eps}{}
\DeclareGraphicsRule{.ps}{eps}{.ps}{}
\DeclareGraphicsRule{.pz}{eps}{.bb}{‘gunzip -c #1}
\DeclareGraphicsRule{.eps.Z}{eps}{.eps.bb}{‘gunzip -c #1}
\DeclareGraphicsRule{.ps.Z}{eps}{.ps.bb}{‘gunzip -c #1}
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{‘gunzip -c #1}
\DeclareGraphicsRule{.ps.gz}{eps}{.ps.bb}{‘gunzip -c #1}
\DeclareGraphicsRule{.pcx}{bmp}{}{}
\DeclareGraphicsRule{.bmp}{bmp}{}{}
\DeclareGraphicsRule{.msp}{bmp}{}{}
\DeclareGraphicsRule{*}{eps}{*}{}
```

The first two commands define the `.eps` and `.ps` extensions as EPS files. The next five commands define extensions for compressed EPS files. The next three commands define extensions for bitmaps (see section 8.5.2). The last command defines any other suffix as an EPS file.

8.3 TeX Search Path and `dvips`

When LaTeX encounters an `\includegraphics` command, it looks in the current directory for the file. If it does not find the file in the current directory, it searches through the TeX path for the file. When the DVI file is converted to PostScript, `dvips` performs the same search routine and everything works well. When the file is an uncompressed EPS file, `dvips` directly includes the specified file. However, when the file is compressed or has a non-EPS format, the file must be operated on by the command specified in `\DeclareGraphicsRule`. For example, the rule

```
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{‘gunzip -c #1}
```

specifies that the `gunzip -c` command be used on files having a `.eps.gz` suffix. Suppose the following command is used

```
\includegraphics{file.eps.gz}
```

If `file.eps.gz` and `file.eps.bb` are in the current directory, LaTeX uses `file.eps.bb` and `dvips` executes `gunzip -c file.eps.gz` to uncompressed the file.

If `file.eps.gz` and `file.eps.bb` are in the directory `/a/b/c/` (on the TeX path), LaTeX searches the path to find `/a/b/c/file.eps.bb`. However, `dvips` does not know what part of the command construction is the filename, and thus cannot find the file `‘gunzip -c file.eps.gz` in the TeX search path. If TeX is using a recent `kpathsea` library (such as the `teTeX` distribution), this problem can be solved by the following graphics rule

```
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}%
                     {‘gunzip -c ‘kpsewhich -n latex tex #1‘}
```

This rule specifies that any file ending with `.eps.gz` is an EPS file whose BoundingBox file ends with `.eps.bb` and is uncompressed with `gunzip -c`. The `kpsewhich -n latex tex` command makes `dvips` look for the compressed file in the directories on the TeX search path. This rule allows `dvips` to operate on any file which TeX can find. This replaces the default definition in `dvips.def` which is essentially

```
\DeclareGraphicsRule {.eps.gz}{eps}{.eps.bb}{‘gunzip -c #1}
```

The rules for other suffixes (such as `.eps.gz` or `.ps.Z`) can be modified similarly.

While the new `\DeclareGraphicsRule` command can be placed at the beginning of every document, it may be more convenient to add the following to the `graphics.cfg` file

```
\AtEndOfPackage{%
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}%
                     {‘gunzip -c ‘kpsewhich -n latex tex #1‘}}
```

and leaving the existing `\ExecuteOptions{dvips}` line.

8.4 The DeclareGraphicsExtensions Command

The `\DeclareGraphicsExtensions` command tells LaTeX which extensions to try if a file with no extension is specified in the `\includegraphics` command. The following graphic extensions are defined by default in `dvips.def`

```
\DeclareGraphicsExtensions{.eps,.ps,.eps.gz,.ps.gz,.eps.Z}
```

With the above graphics extensions specified, `\includegraphics{file}` first looks for `file.eps`, then `file.ps`, then `file.eps.gz`, etc. until a file is found. The `\DeclareGraphicsExtensions` command allows the graphics to be specified with

```
\includegraphics{file}
```

instead of

```
\includegraphics{file.eps}
```

The first syntax has the advantage that if you later decide to compress `file.eps`, you need not edit the LaTeX file.

8.5 Non-EPS Graphic Files

While it is easy to insert EPS graphics into LaTeX documents, it is not as straight-forward to insert non-EPS graphics (GIF, TIFF, JPEG, PICT, etc.). A simple solution is to determine whether the application which generated the non-EPS graphic also generates EPS output. If not, a graphics-conversion program must be used to convert the graphics to PostScript. See

```
http://www.wizards.dupont.com/cristy/ImageMagick.html
```

for information on ImageMagick, a very good graphics-conversion utility for Unix, Linux, and VMS that is distributed for free from `ftp.wizards.dupont.com` and other sites. See

```
http://www.sun.com/sunsoft/catlink/xv/note.html
```

for information on `xv`, an X-Windows graphics viewing and conversion program distributed from `ftp.cis.upenn.edu` as shareware. Unlike ImageMagick, `xv` does not provide command-line conversion capabilities for on-the-fly graphics conversion.

Since a non-EPS graphics file may be smaller than the corresponding EPS file, it may be desirable to keep the graphics in a non-EPS format and convert them to PostScript when the DVI file is converted to PostScript. If `dvips` is used, this on-the-fly conversion can be specified by the command option in `\DeclareGraphicsRule`. For example, using on-the-fly conversion to insert `file2.gif` into a LaTeX document requires the following steps

1. Find a GIF-to-PS conversion program (assume it's called `gif2ps`)
2. One needs to create a BoundingBox file which specifies the natural size of the `file2.gif` graphics. To do this, convert `file2.gif` to PostScript and
   1. If the Postscript file is EPS, save the BoundingBox line in `file2.gif.bb`
   2. If the Postscript file is not EPS, determine the appropriate BoundingBox (see section 3) and place those numbers in a `%%BoundingBox:` line in `file2.gif.bb`
3. Keep `file2.gif` and `file2.gif.bb` and delete the PostScript file.
4. Include `\DeclareGraphicsRule{.gif}{eps}{.gif.bb}{‘gif2ps #1}` before the `\includegraphics` command in the LaTeX file.

When `\includegraphics{file.gif}` is issued, LaTeX reads the BoundingBox from `file.gif.bb` and tells `dvips` to use `gif2ps` to convert `file.gif` to EPS.

8.5.1 GIF Example

While the commands necessary for including non-EPS graphics are dependent on the operating system and the graphics conversion program, this section provides examples for two common Unix conversion programs. The commands

```
\DeclareGraphicsRule{.gif}{eps}{.gif.bb}{‘convert #1 ’eps:-’ }
\begin{figure}
   \centering
   \includegraphics[width=3in]{file2.gif}
   \caption{GIF Graphic}
\end{figure}
```

use the `convert` program (part of the ImageMagick package) package to translate the GIF file into EPS. The command

```
convert file2.gif ’eps:-’
```

translates `file2.gif` into EPS format (specified by the "`eps:`" option), sending the result to standard output (specified by the "`-`" specification).

Alternatively, one can use the `ppm` utilities in which `giftoppm`, `ppmtopgm`, and `pgmtops` convert the GIF file to EPS via the `ppm` and grayscale `pgm` formats. In Unix, the piping between these programs is specified by the following `\DeclareGraphicsRule` command

```
\DeclareGraphicsRule{.gif}{eps}{.gif.bb}{‘giftoppm #1 | ppmtopgm | pgmtops}
```

8.5.2 Direct Support for Non-EPS Graphics

It is often requested that LaTeX and `dvips` support the direct inclusion of non-EPS graphic formats, making it as easy as inserting EPS files. While this would be convenient, there unfortunately are problems with this.

For example, most non-EPS graphic formats use binary files which cannot be read by TeX, which prevents LaTeX from determining the size of the non-EPS graphics. Furthermore, supporting non-EPS graphics would also require `dvips` to incorporate graphics-conversion capabilities (GIF-to-PS, TIFF-to-PS, etc.). This would not only require a lot of programming, it would also require more future maintenance.

Rather than directly incorporating graphics-conversion routines, `dvips` provides a mechanism of calling external conversion programs. This mechanism can be accessed from LaTeX by using the command argument of `\DeclareGraphicsRule`. This has the benefit of being more flexible than direct support, and since it keeps the graphics-conversion uncoupled from the DVI-to-PS conversion, users are free to choose their own graphics-conversion program.

While LaTeX and `dvips` generally do not support the direct inclusion of non-EPS graphics, there are some exceptions

1. If `dvips` is compiled with `-Demtex`, it supports some EmTeX `\special` commands, allowing it to include PCX, BMP, or MSP bitmaps.
2. Some commercial versions of LaTeX support non-EPS graphics
   1. Textures for the Macintosh supports PICT graphics.
   2. Y&Y's TeX package for Windows includes the DVI-to-PS converter DVIPSONE which supports TIFF files. However, TeX cannot read the binary TIFF files, preventing LaTeX from reading the TIFF tags the same way it reads EPS BoundingBox information. Since LaTeX cannot determine the natural size of TIFF graphics, the user must still use a `.bb` file or specify the `bb` parameters explicitly in the `\includegraphics` command.

   Check your documentation or contact the company's customer service for the correct syntax.

9 The PSfrag System

While there are many drawing and analysis packages which produce EPS files, most of them do not support symbols and equations as well as LaTeX . The PSfrag system allows LaTeX users to replace text strings in EPS files with LaTeX text or equations. Currently available for both DOS and Unix, the PSfrag system consists of the LaTeX style file `psfrag.sty` and the `perl` script `ps2frag` and is well-documented by [7].

PSfrag currently does not support compressed or non-EPS graphics. This means that if PSfrag is used for even one graphic in a document, all of the document's graphics must be non-compressed EPS files.

The procedure for using `PSfrag`

1. Create an EPS file.
2. At the operating system prompt, type `ps2frag file.eps` which scans the EPS file `file.eps` for text strings and then records these locations in the EPS file. Since this added information is in the form of header comments in the EPS file, it does not change the appearance of the EPS output.

3. In the LaTeX document, use the following commands
   1. Include `\usepackage{psfrag}` in the preamble.
   2. Use the `\psfrag` command to specify the EPS text and the LaTeX string to replace it. This makes the specified substitution occur in any subsequent `\includegraphics` command issued in the same environment.
   3. Use the `\includegraphics` command as usual.

The LaTeX `\psfrag` command has the following syntax

```
\psfrag{PStext}[posn][PSposn][scale][rot]{text}
```

with its arguments described in Table 5.

`PStext`	Text in EPS file to be replaced. PSfrag is sensitive about what type of text it replaces. For example, if the EPS file contains the text `Error (%)`, the percent sign confuses LaTeX and PSfrag *cannot* be used on the file, regardless of whether PSfrag replaces `Error (%)`. Instead, regenerate the EPS file using text such as Error (percent) which does not contain any of the LaTeX special characters.
`posn`	*(Optional, Defaults to `[Bl]`.)* Position of placement point relative to new LaTeX text. `[ ]` indicates center.
`PSposn`	*(Optional, Defaults to `[Bl]`.)* Position of placement point relative to existing EPS text. `[ ]` indicates center.
`scale`	*(Optional, defaults to 1.)* Scaling factor for the text. For best results, avoid using the scaling factor and instead use LaTeX type-size commands such as `\small` and `\large`
`rot`	*(Optional, defaults to zero.)* When this rotation angle is zero, the new text is inserted at the same angle as the existing EPS text. When an angle is specified here, it is the angle of rotation of the new text relative to the existing text. The angle is in degrees with a counterclockwise rotation being positive. This option is useful when dealing with EPS files generated by applications which only allow horizontal text. This option effectively adds rotated-text capabilities to those applications.
`text`	The LaTeX text to insert into the EPS graphic. Like regular LaTeX text, math formulas must be enclosed by dollar signs (e.g., `$\frac{1}{2}$` or `$x^2$`) and special symbols can be used (e.g., `\%` produces %).

Table 5: `psfrag` Options

The `posn` and `PSposn` options are one of the 12 points shown in Figure 3 on page 79, except that the `c` specifier is not available (e.g., to align the left-center, use `[l]` instead of `[lc]`; to align centers, use `[ ]` instead of `[cc]`). See [7] for examples of various combinations of placement points.

9.1 PSfrag Example

The commands

```
\includegraphics{pend.eps}
```

include the graphic without any PSfrag replacement, producing Figure 4. The commands

```
\psfrag{q1}{$\theta_1$}
\psfrag{q2}{$\theta_2$}
\psfrag{L1}{$L_1$}
\psfrag{L2}{$L_2$}
\psfrag{P1}[][]{$P_1$}
\psfrag{P2}[][]{\Large $P_2$}
\includegraphics{pend.eps}
```

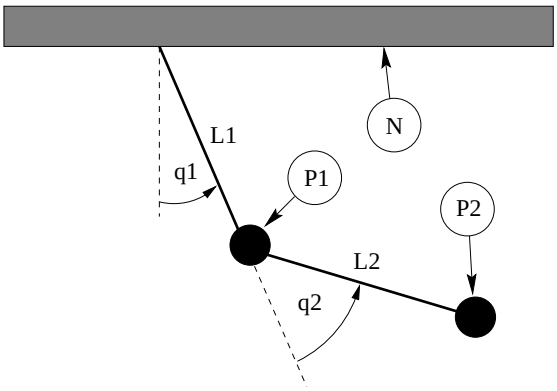


Figure 4: Without PSfrag Replacement

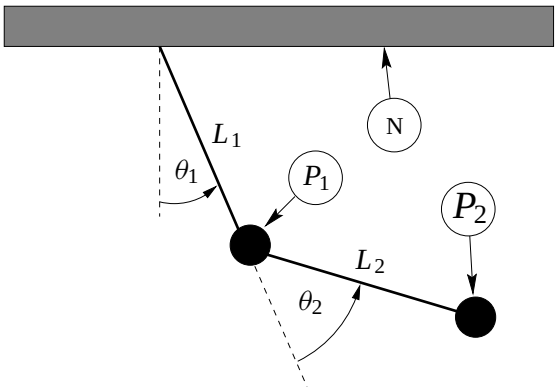


Figure 5: With PSfrag Replacement

include the graphic with PSfrag replacement, producing Figure 5. The first four `\psfrag` commands position the new LaTeX text such that its left baseline point corresponds to the left baseline point of the EPS text. The last two `\psfrag` commands position the new LaTeX text such that its center corresponds to the center of the EPS text.

Note that one need not replace all the EPS text with LaTeX text. For example, the `N` tag is left unchanged in Figure 5. Also note that `\psfrag` matches *entire* text strings. Thus the command

```
\psfrag{pi}{$\pi$}
```

replaces the string `pi` with π, but does not affect the strings `pi/2` or `2pi`. Separate `\psfrag` commands must entered for these strings.

9.2 LaTeX Text in EPS File

In the previous section, the `\psfrag` command specified the LaTeX text in the LaTeX file. While this is the most popular method, PSfrag's `\tex` command includes LaTeX text directly in EPS files. The `\tex` command has the following syntax

```
\tex[posn][PSposn][scale][rot]{text}
```

which is the same as `\psfrag` command, except there is no `{PStext}` argument. Unlike the `\psfrag` command, the `\tex` command is placed in the EPS file.

For example, if an EPS file contains the text `\tex{$x^2$}` PSfrag automatically replaces it with x^2. The left-baseline point of x^2 is aligned with the left-baseline point of `\tex{$x^2$}`. Note that PSfrag does the replacement automatically; apart from the `\usepackage{psfrag}` command, it does not require any commands in the LaTeX file. Placement, scaling, and rotation options can be specified as with the `\psfrag` command. If an EPS file contains the text `\tex[][]{$x^2$}` PSfrag replaces it with a centered x^2. The `\tex` command must be *entire* text string. For example, the text

```
Transfer Function \tex{$\frac{s+a}{s+b}$}
```

produces an error. Instead use

```
\tex{Transfer Function $\frac{s+a}{s+b}$}
```

The advantage of the `\tex` command is that the LaTeX file doesn't need to be edited when an EPS file is modified. The `\tex` command has two disadvantages. First, the EPS file cannot be used for non-LaTeX purposes, while the EPS graphic in Figure 4 could be used without replacement. Second, if `\tex` command contains complicated formulas, the text can extend beyond the edge of the graphics, enlarging the EPS BoundingBox. This oversized BoundingBox causes incorrect graphic placement in LaTeX .

9.3 Text Scaling in PSfrag

A subtlety of the `\includegraphics` command (see [3, page 3]) comes into play with PSfrag. When scaling options are specified *before* rotation

```
\includegraphics[width=3in,angle=30]{file.eps}
```

the scaling is implicitly handled by the graphics inclusion function. However, when scaling options are specified *after* rotation

```
\includegraphics[angle=30,width=3in]{file.eps}
```

the graphic is first included at its natural size, then rotated, and then scaled. Since PSfrag replaces the new text during the graphics inclusion, the second command scales the new PSfrag text while the first command does *not*. When the included size of the EPS graphic greatly differs from its natural size, the two commands produce very different results. See [7, pages 10-11] for information.

10 Graphics in Page Header or Footer

The fancyhdr package (an improved version of the old fancyheadings package) makes it easy to customize a document's page headers and footers. It is often desired to place a logo or other EPS graphics in the header or footer, which results in the same EPS file being included multiple times.

10.1 Including An EPS File Multiple Times

There are four common methods for including the same EPS graphics many times

1. Using `\includegraphics{file.eps}` wherever you want the graphic has two problems
   1. LaTeX must find and read the file every time `\includegraphics` is used.
   2. The EPS graphics commands are repeated in the final PS file, producing a large file.
3. Save the graphics in a LaTeX box, using the box wherever you want the graphic. This saves LaTeX time since it must only find and read the file once. However, it does not reduce the size of the final PostScript file.
   At the beginning of the file, include the following commands
   ```
   \newsavebox\mygraphic
   \sbox\mygraphic{\includegraphics{file.eps}}
   ```
   Then use the command `\usebox{\mygraphic}` wherever you want the graphic.
4. Define a PostScript command which draws the graphics, and then issue the Postscript command wherever you want the graphic. Section 10.2 describes this procedure.
   Since the final PostScript file includes the graphics commands only once, the final PostScript file is much smaller. Note that since the graphics commands are stored in printer memory while the final PostScript file is being printed, this method may cause the printer to run out of memory and not print the document.
   Although this method results in a small final PostScript file, it still requires LaTeX to find and read the file containing the PostScript commands.
5. Like the previous method, define a PostScript command which draws the graphics, but include this command in a LaTeX box. This results in a small final PostScript file and only requires LaTeX to find and read the file once.

10.2 Defining a PostScript Command

To convert the EPS graphics into a PostScript Command, the EPS file must be broken into two files, one which defines the PostScript dictionary and the graphics commands, and another which includes the header information and the uses the previously-defined PostScript command. For example, an EPS file created by xfig has the form

```
%!PS-Adobe-2.0 EPSF-2.0
%%Title: /tmp/xfig-fig017255
%%Creator: fig2dev Version 2.1.8 Patchlevel 0
%%CreationDate: Sun Sep  3 15:36:01 1995
%%Orientation: Portrait
%%BoundingBox: 0 0 369 255
%%Pages: 0
%%EndComments
/$F2psDict 200 dict def
$F2psDict begin
...
%%EndProlog

$F2psBegin
...
$F2psEnd
```

Where `...` indicates unlisted commands. The EPS file generally contains three parts

1. The header commands which begin with `%`
2. The Prolog section which starts with
   ```
   /$F2psDict 200 dict def
   ```
   and ends with
   ```
   %%EndProlog
   ```
   The Prolog defines the commands in the PostScript dictionary used by the EPS file. In this example, the dictionary is named `$F2psDict` although other names can be used.
3. The last part contains the commands used to draw the graphics.

Suppose the above EPS file is named `file.eps`. Create the files `file.h` and `file.ps` where `file.h` contains

```
/$F2psDict 200 dict def
```

```
$F2psDict begin
...
%%EndProlog

/MyFigure {
$F2psBegin
...
$F2psEnd
} def
```

and `file.ps` contains

```
%!PS-Adobe-2.0 EPSF-2.0
%%Title: /tmp/xfig-fig017255
%%Creator: fig2dev Version 2.1.8 Patchlevel 0
%%CreationDate: Sun Sep  3 15:36:01 1995
%%Orientation: Portrait
%%BoundingBox: 0 0 369 255
%%Pages: 0
%%EndComments
$F2psDict begin MyFigure end
```

`file.h` defines the dictionary and defines the PostScript command `/MyFigure`, while `file.ps` contains the header information and uses the PostScript command defined in `file.h`. In particular, it is important that the `file.ps` header includes the `%!PS...` line and the BoundingBox line. The graphics can then be used in the LaTeX document as

```
\documentclass{article}
\usepackage{graphicx}
...
\special{header=file.h}
...
\begin{document}
   ...
   \includegraphics[width=2in]{file.ps}
   ...
   \includegraphics[totalheight=1in]{file.ps}
   ...
\end{document}
```

Note that the original file `file.eps` is not used. Since the graphics commands in `file.h` are only included once, the final PostScript file remains small. However, this still requires LaTeX to find and read `file.ps` whenever the graphics are used. The following commands save the graphics in a LaTeX box to produce a small final PostScript file while reading `file.ps` only once.

```
\documentclass{article}
\usepackage{graphicx}
...
\special{header=file.h}

\newsavebox\mygraphic
\sbox\mygraphic{\includegraphics[width=2in]{file.ps}}

\begin{document}
   ...
   \usebox{\mygraphic}
   ...
   \resizebox*{1in}{!}{\usebox{\mygraphic}}
   ...
\end{document}
```

Like the previous example, these commands produce a 2-inch wide graphic and another graphic whose totalheight is 1 inch.

10.3 fancyhdr Package

An easy method of including graphics in the heading is to use the fancyhdr package, which is documented by [8]. The header consists of three parts: its left field, its center field, and its right field. The `\fancyhead` command specifies the contents of the header fields, with the `L,C,R` options specifying which field(s) the command modifies. For example

```
\pagestyle{fancy}
\fancyhead[C]{My Paper}
```

causes the center header field to be “My Paper”, while

```
\pagestyle{fancy}
\fancyhead[L,R]{\textbf{Confidential}}
```

causes the left and right header fields to be “**Confidential**”. If no `L,C,R` option is specified, it applies to all three header fields. Thus `\fancyhead{}` is used to clear all the header fields. The `\fancyfoot` command similarly specifies the left, center, and right footer fields.

Note that the above `\fancyhead` commands only apply to pages whose style are “fancy”. Even though the command `\pagestyle{fancy}` causes the document to have a fancy page style, some pages (title pages, table of contents pages, the first page of chapters, etc.) are still given a plain pagestyle by default.

Graphics in Page Header/Footer

The commands in the fancyhdr package can insert graphics in the headers and footers. For example, after splitting the EPS file `file.eps` into the two file `file.h` and `file.ps` as described in section 10.2, the commands

```
\documentclass{article}
\usepackage{fancyhdr,graphicx}

\renewcommand{\headheight}{0.6in}  %% must be large enough for graphic
\renewcommand{\textheight}{7.5in}

% Define PostScript graphics command
\special{header=file.h}

% Save graphics in LaTeX box
\newsavebox\mygraphic
\sbox\mygraphic{\includegraphics[totalheight=0.5in]{file.ps}}

\pagestyle{fancy}
\fancyhead{}   % clear all header fields
\fancyhead[L]{\usebox{\mygraphic}}
\fancyfoot{}   % clear all footer fields
\fancyfoot[C]{\thepage}
\renewcommand{\headrulewidth}{0.5pt}
\renewcommand{\footrulewidth}{0pt}

\begin{document}
...
\end{document}
```

places the graphics at the top left of each “fancy” page with a 0.5 pt horizontal line drawn under the header. Additionally, the page number is placed at the bottom center of each page, with no horizontal line drawn above the footer.

Odd/Even Headings

When the `[twoside]` documentclass option is used, one may want to individually specify the odd and even page headers/footers. The `E,O` `\fancyhead` options specify the even and odd page headers, respectively. If the `E,O` options are not specified, the command applies to both even and odd pages. Likewise the `E,O` `\fancyfoot` options specify the even and odd page footers. For example,

```
\pagestyle{fancy}
\fancyhead[LE]{My Paper}
\fancyhead[RO]{My Name}
\fancyfoot[C]{\thepage}
```

places “My Paper” in the upper left of even fancy pages, “My Name” in the upper right of odd fancy pages, and the page number in the bottom center of all fancy pages. Replacing the

```
\fancyhead[L]{\usebox{\mygraphic}}
```

command in the above example with

```
\fancyhead[LE,RO]{\usebox{\mygraphic}}
```

places the graphic at the top outside (the left side of even pages, right side of odd pages) of all fancy pages.

Modifying Plain Pages

Although the above commands do not affect pages with plain pagestyles, the `\fancypagestyle` command can be used to modify the plain pagestyle. For example

```
\documentclass{article}
\usepackage{fancyhdr,graphicx}

\renewcommand{\headheight}{0.6in}  %% must be large enough for graphic
\renewcommand{\textheight}{7.5in}

% Define PostScript graphics command
\special{header=file.h}

% Save graphics in LaTeX box
\newsavebox\mygraphic
\sbox\mygraphic{\includegraphics[totalheight=0.5in]{file.ps}}

\pagestyle{fancy}
\fancyhead{}   % clear all header fields
\fancyhead[L]{\usebox{\mygraphic}}
\fancyfoot{}   % clear all footer fields
\fancyfoot[C]{\thepage}
\renewcommand{\headrulewidth}{0.5pt}
\renewcommand{\footrulewidth}{0pt}

\fancypagestyle{plain}{%
   \fancyhead{}   % clear all header fields
   \fancyhead[L]{\usebox{\mygraphic}}
   \fancyfoot{}   % clear all footer fields
   \fancyfoot[C]{\thepage}
   \renewcommand{\headrulewidth}{0.5pt}
   \renewcommand{\footrulewidth}{0pt}}

\begin{document}
...
\end{document}
```

place the graphic at the upper left of every page (both plain and fancy). Likewise, when the `twoside` documentclass option is used, replacing both of the

```
\fancyhead[L]{\usebox{\mygraphic}}
```

commands with

```
\fancyhead[LE,RO]{\usebox{\mygraphic}}
```

places the graphic at the top outside of every page (both plain and fancy).

Part IV

Related LaTeX Commands

11 The figure Environment

Graphics can be inserted as part of a LaTeX `\figure` environment, which allows the graphics to float for better formatting, especially for large graphics. The `\figure` environment also makes it easy to reference the graphic. The commands

```
\begin{figure}[htb]
   \centering
   \includegraphics[totalheight=2in]{graph.eps}
   \caption{This is an inserted EPS graphic}\label{fig:graph}
\end{figure}

The graph in Figure~\ref{fig:graph} is
from an EPS file generated by gnuplot.
```

insert the graphic in a figure and place a caption under the graphic. The optional `\label` command specifies a label which is used by the `\ref` command to reference the figure (the `\label` command must be *after* the `\caption` command). Note that the figure environment can only be used in *outer paragraph mode* and thus cannot be used inside any box (such as parbox or minipage).

11.1 Caption Vertical Spacing

While the figure caption is usually placed below the graphic, it can be placed above the graphics simply by placing the `\caption` command before the graphics-inclusion command. For example, the commands

```
\begin{figure}[htb]
   \centering
   \caption{Caption Above Graphic}
   \includegraphics[width=1in]{box.eps}
\end{figure}
```

produce Figure 6.

Figure 6: Caption Above Graphic

Since captions are generally placed below the graphic, LaTeX places more vertical spacing above the caption than below it. As a result, the caption in Figure 6 is placed quite close to the graphic. The spacing above and below the caption is controlled by the two lengths `\abovecaptionskip` (which is `10pt` by default) and `\belowcaptionskip` (which is zero by default). The standard LaTeX commands `\setlength` and `\addtolength` are used to modify these lengths. The commands

```
\setlength{\abovecaptionskip}{5pt}
\setlength{\belowcaptionskip}{0.5cm}
```

provides a 5 point spacing above the caption and a 0.5 centimeter spacing below the caption. The commands

```
\addtolength{\abovecaptionskip}{5pt}
\addtolength{\belowcaptionskip}{-5pt}
```

increases the spacing above the captions by 5 points and decreases the spacing below the captions by 5 points. For example, the commands

```
\begin{figure}[htb]
   \setlength{\belowcaptionskip}{10pt}
   \centering
   \caption{Caption Above Graphic}
   \includegraphics[width=1in]{box.eps}
\end{figure}
```

produce Figure 7.

Figure 7: Caption Above Graphic

11.2 Figure Placement Options

LaTeX figures are "floats" whose placement are decided by LaTeX. Since your taste in figure-placement may differ from that of LaTeX, the `\figure` environment has placement options

- **h** *Here:* Place the figure in the text where the figure command is located.
- **t** *Top:* Place the figure at the top of a page.
- **b** *Bottom:* Place the figure at the bottom of a page.
- **p** *Page of Floats:* Place the figure on a separate page which contains only floats.

The placement options in the above example are `[htb]` which means that LaTeX first tries to place the figure at that location, then tries to place the figure at the top of a page, and finally tries to place the figure at the bottom of a page. When LaTeX "tries" to place a figure, it checks how many figures are already on the page and other esthetic concerns. If LaTeX determines that the figure wouldn't look good, it tries the next placement option.

The order in which the placement options are specified does *not* make any difference. The placement options are attempted in the order `h-t-b-p` regardless of the order in which the options are specified. Thus `[hb]` and `[bh]` are both attempted as `h-b`.

To make LaTeX "try really hard" in its float placement, specify an exclamation point in the placement options (e.g., `\begin{figure}[!ht]`) which makes LaTeX suspend its esthetic rules and do its best to make the requested placement. Even with the `!` option, LaTeX has the final say in the placement and reserves the right to override the request. For example, if the commands

```
\begin{figure}[!ht]
   \includegraphics[totalheight=4in]{graph.eps}
\end{figure}
```

occur 3 inches from the bottom of the page, LaTeX objects to leaving 3 inches of whitespace at the bottom of the page and overrides the `[!h]`, with the text which is after the figure in the .tex file filling the bottom 3 inches of the page.

If you feel LaTeX is making poor float placement decisions, you may need to tweak its placement algorithm by modifying the float parameters (see [4, pages 199-200], [5, pages 141-143], or [6, pages 174-175]).

11.2.1 The oat Package's `[H]` Placement Option

The oat package adds an `[H]` option to the `\figure` environment which *always* places the float "here". However, the `[H]` option should generally be avoided, as the `[!ht]` option is a better way of producing the desired behavior.

To use the `[H]` option, include a `\usepackage{float}` in the preamble and issue the `\restylefloat{figure}` command *before* the `\begin{figure}[H]` command is used (See [5, page 149]). When using the `[H]` option, the user is responsible for managing the document to avoid large sections of whitespace.

While the figure environment defined by the oat package allows the `[H]` option, it also places the figure caption below the figure environment. While this does not affect simple figures, it prevents captions above graphics as in Figure 6 or the construction of side-by-side and other complex figure arrangements.

12 Landscape Figures

In a document with portrait orientation, there are three methods for producing figures with landscape orientation.

1. The lscape package provides a `landscape` environment, which treats the left edge of the paper as the top of the page, causing any text, tables, or figures in the `landscape` environment to have landscape orientation.
2. The rotating package provides a `sidewaysfigure` environment which is similar to the `figure` environment except that the figures have landscape orientation.
3. The rotating package provides a `\rotcaption` command which is similar to the `\caption` command except that caption has landscape orientation.

Differences between methods

- Both options 1 and 2 place the rotated figure on a separate page. Option 3 produces an individual float which need not be on its own page.
- The full-page figure produced by Option 2 will float to provide better document formatting. Since the figure(s) produced by Option 1 can only float within the landscape pages, it may result in a partially-empty page before the figure.
- The `landscape` environment in Option 1 can be used to produce landscape pages containing any combination of text, tables, and figures. Option 2 produces only rotated figures.

12.1 Landscape Environment

The lscape package (which is part of the standard "graphics bundle" distributed with LaTeX) defines the `landscape` environment, which provides a method of placing landscape pages in a portrait document. The landscape pages are rotated such that the left edge of the portrait page is the top edge of the landscape page.

Entering `\begin{landscape}` prints all unprocessed portrait floats and then switches to landscape orientation. Likewise, `\end{landscape}` prints all unprocessed landscape floats and then switches back to portrait orientation.

The entire contents of the `landscape` environment is typeset with landscape orientation. This may include any mixture of text, figures, and tables. If the landscape environment contains only a figure environment

```
\begin{landscape}
  \begin{figure}
     \centering
     \includegraphics[width=4in]{box.eps}
     \caption{Landscape Figure}
  \end{figure}
\end{landscape}
```

the `landscape` environment produces a landscape figure. Note that since the `landscape` environment starts a new page, it may result in a partially-blank page.

12.2 Sidewaysfigure Environment

The rotating package provides the `sidewaysfigure` environment which produces figures with landscape orientation. For example

```
\begin{sidewaysfigure}
   \centering
   \includegraphics[width=4in]{box.eps}
   \caption{Sidewaysfigure Figure}
\end{sidewaysfigure}
```

produces Figure 8.

Unlike the `landscape` environment, the figure produced by `sidewaysfigure` can float within the portrait pages to avoid the partially-blank page that the `landscape` environment may produce. (The rotating package also provides a `sidewaystable` environment for producing tables with landscape orientation.) Note that the `landscape` environment is much more flexible, allowing the landscape pages to consist of a mixture of text, tables, and figures.

The default orientation of the figures produced by `sidewaysfigure` depends on whether the document is processed with the `oneside` or `twoside` documentclass option

- When the `oneside` option is chosen, the bottom of graphic is towards the the right edge of the portrait page.
- When the `twoside` option is chosen, the bottom of graphic is towards the the outside edge of the portrait page.

This default behavior can be overridden by options to the `\usepackage{rotating}` command.

```
\usepackage[rotateleft]{rotating}
```

causes the bottom of the `sidewaysfigure` graphics to be towards the left edge of the portrait page (regardless of `oneside` or `twoside` options). Similarly,

```
\usepackage[rotateright]{rotating}
```

causes the bottom of the `sidewaysfigure` graphics to be towards the right edge of the portrait page.

12.3 Rotcaption Command

The methods in Sections 12.1 and 12.2 both produce full-page landscape figures, which may not be necessary for smaller landscape figures. The rotating package's `\rotcaption` command can be used to construct smaller landscape figures. For example

```
\begin{figure}
  \centering
  \begin{minipage}[c]{1in}
     \includegraphics[angle=90,width=\textwidth]{box.eps}
  \end{minipage}
  \begin{minipage}[c]{0.5in}
     \rotcaption{Rotcaption Caption}
     \label{fig:rotcaption}
  \end{minipage}
\end{figure}
```

produces Figure 9. The caption produced by `\rotcaption` is always rotated such that its bottom is towards the right edge of the paper. Unlike the methods in Sections 12.1 and 12.2, the `\rotcaption` command does not rotate the graphics. Therefore, the `\includegraphics` command in the above example required the `angle=90` option.

13 Side-by-Side Graphics

The commands necessary for side-by-side graphics depend on how the user wants the graphics organized. This section covers three common methods of organizing side-by-side graphics

1. The side-by-side graphics are combined into a single figure.
2. The side-by-side graphics each form their own figure (e.g., Figure 12, Figure 13, etc.)
3. The side-by-side graphics each form a subfigure (e.g., Figure 12a, Figure 12b, etc.) of a single figure (Figure 12).

While this section specifically discusses side-by-side graphics, most of the information is also valid for vertically-stacked graphics and complex figures such as Figures 33-39 on Page 110.

13.1 Side-by-Side Graphics in a Single Figure

The two most common methods for placing side-by-side graphics in a figure are

1. Multiple `\includegraphics` commands
2. Multiple `minipage` environments, each of which contain a `\includegraphics` command

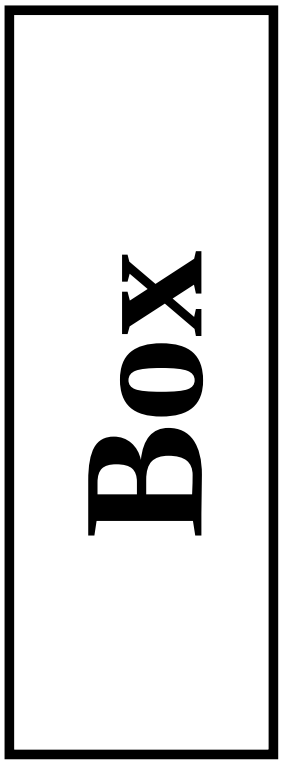


Figure 8: Sidewaysfigure Figure

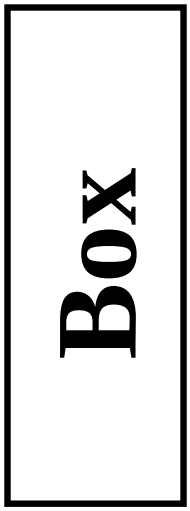


Figure 9: Rotcaption Caption

13.1.1 Side-by-Side includegraphics Commands

While spacing side-by-side graphics in a figure is as simple as

```
\begin{figure}
   \centering
   \includegraphics[width=1in]{file1.eps}
   \includegraphics[width=2in]{file2.eps}
   \caption{Two Graphics in One Figure}
\end{figure}
```

there usually are horizontal-spacing commands such as `\hspace{1in}` or `\hfill` between the `\includegraphics` commands. For example

```
\begin{figure}
   \centering
   \includegraphics[width=1in]{box.eps}%
   \hspace{1in}%
   \includegraphics[width=2in]{box.eps}
   \caption{Two Graphics in One Figure}
\end{figure}
```

produces Figure 10 which is 4 inches wide (1 inch for `file1.eps`, 1 inch for the `\hspace`, and 2 inches for `file2.eps`). This 4-inch-wide figure is centered on the page. If `\hfill` is used instead of `\hspace`, the graphics are pushed to the margins.

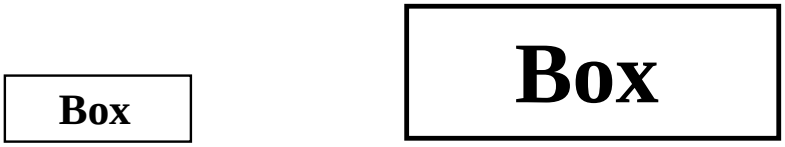


Figure 10: Two Graphics in One Figure

13.1.2 Side-by-Side Minipage Environments

Placing the `\includegraphics` commands inside `minipage` environments provides the user more control over the graphics' horizontal and vertical placement. For example

```
\begin{figure}
   \centering
   \begin{minipage}[c]{0.5\textwidth}
      \centering \includegraphics[width=1in]{box.eps}
   \end{minipage}%
   \begin{minipage}[c]{0.5\textwidth}
      \centering \includegraphics[width=2in]{box.eps}
   \end{minipage}
   \caption{Centers Aligned Vertically}
\end{figure}
```

Figure 11: Centers Aligned Vertically

produces Figure 11.

Notes on this example

- Like any other LaTeX object, minipages are positioned such that their baseline is aligned with the current baseline. The minipage `[c]` option defines the minipage's baseline as its centerline. The `[b]` option defines the minipage's baseline as the baseline of the bottom line of the minipage (which is not necessarily the bottom of the minipage). The `[t]` option defines the minipage's baseline as the baseline of the top line of the minipage (which is not necessarily the top of the minipage). See section 14 for information on the `minipage` environment and its placement options.
- The `%` after the first `\end{minipage}` command prevents a space from being inserted between the `minipage` boxes. Such a space would use some horizontal space, preventing both minipages from fitting on the same line.

When the widths of the minipages do not add to `1.0\textwidth`, the `\hspace` or `\hfill` commands can be used to specify to horizontal spacing. For example

```
\begin{figure}
   \centering
   \begin{minipage}[c]{1in}
      \centering \includegraphics[width=\textwidth]{box.eps}
   \end{minipage}
   \hspace{1in}
   \begin{minipage}[c]{2in}
      \centering \includegraphics[width=\textwidth]{box.eps}
   \end{minipage}
   \caption{Centers Aligned Vertically}
\end{figure}
```

produces a figure with the same horizontal spacing as Figure 10, but the centers of the boxes are aligned vertically.

13.2 Side-by-Side Figures

In the previous section, multiple `minipage` environments were used inside a `figure` environment to produce a single figure consisting of multiple graphics. Placing `\caption` statements inside the minipages makes the minipages themselves become figures. For example

```
\begin{figure}
   \begin{minipage}[b]{0.5\linewidth}
      \centering \includegraphics[width=1in]{box.eps}
      \caption{Small Box}\label{fig:side:a}
   \end{minipage}%
   \begin{minipage}[b]{0.5\linewidth}
     \centering \includegraphics[width=1.5in]{box.eps}
     \caption{Big Box}  \label{fig:side:b}
   \end{minipage}
\end{figure}
```

produces Figures 12 and 13.

Figure 12: Small Box

Figure 13: Big Box

Although the above commands include *one* figure environment, the commands produce *two* figures. Since the `\caption` command actually produces the figure, figure environments with multiple `\caption` commands produce multiple figures.

13.2.1 Alignment Problems with Side-by-Side Figures

The `[b]` options aligned the bottoms of Figures 12 and 13. However, long captions may affect this alignment. For example

```
\begin{figure}
   \begin{minipage}[b]{.333\linewidth}
      \centering \includegraphics[width=1in]{box.eps}
      \caption{Small Box with a Long Caption}\label{fig:side:c}
   \end{minipage}%
   \begin{minipage}[b]{.333\linewidth}
      \centering \includegraphics[width=1.5in]{box.eps}
      \caption{Medium Box}\label{fig:side:d}
   \end{minipage}%
   \begin{minipage}[b]{.333\linewidth}
      \centering \includegraphics[width=2.0in]{box.eps}
      \caption{Big Box}\label{fig:side:e}
   \end{minipage}
\end{figure}
```

produces Figures 14, 15, and 16.

Figure 14: Small Box with a Long Caption

Figure 15: Medium Box

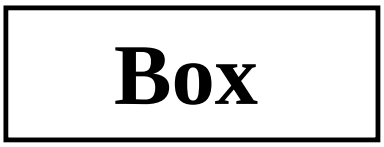


Figure 16: Big Box

The long caption of Figure 14 makes it unaligned with the other figures. In this case, the baselines of all the figures are their bottoms, so the alignment can be corrected by changing the `minipage` positioning option from `[b]` to `[t]` which aligns the baselines of the graphics (see Section 14 for information). If the baselines of the graphics do not correspond to their bottoms, the `[t]` option does not produce the desired positioning. Instead, invisible vertical lines (called *struts*) can be placed in the captions of the other figures to make LaTeX think that all the captions are two lines long.

```
\begin{figure}
   \begin{minipage}[b]{.333\linewidth}
      \centering \includegraphics[width=1in]{box.eps}
      \caption{Small Box with a Long Caption}\label{fig:side:cc}
   \end{minipage}%
   \begin{minipage}[b]{.333\linewidth}
      \centering \includegraphics[width=1.5in]{box.eps}
      \caption[Medium Box]
          {Medium Box \protect\rule[-\baselineskip]{0pt}{2\baselineskip}}
       \label{fig:side:dd}
   \end{minipage}%
   \begin{minipage}[b]{.333\linewidth}
      \centering \includegraphics[width=2.0in]{box.eps}
      \caption[Big Box]
          {Big Box \protect\rule[-\baselineskip]{0pt}{2\baselineskip}}\label{fig:side:ee}
   \end{minipage}
\end{figure}
```

which produces Figures 17, 18, and 19.

Figure 17: Small Box with a Long Caption

Figure 18: Medium Box

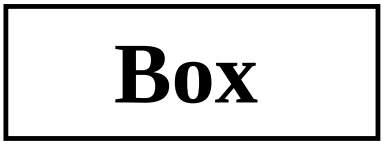


Figure 19: Big Box

The command `\rule[start]{width}{height}` produces an vertical line with a width of `width` starting `start` above the baseline and with a height `height`. When the width is zero, the line becomes invisible and is called a *strut.* In the above captions, the strut

```
\rule[-\baselineskip]{0pt}{2\baselineskip}}
```

starts one line below the baseline and continues to the top of the current line. This makes LaTeX think that, like the Figure 17 caption, the captions for Figures 18 and 19 are two lines tall. Since the `\rule` command is fragile, the `\protect` command must be used so `\rule` can be used in the `\caption` command. The `\caption[Big Box]` option specifies that the text "Big Box" should be used in the list of figures (where the extra vertical space is not desired).

13.3 Side-by-Side Subfigures

It is often desirable to refer to side-by-side graphics both individually and as a group. The `\subfigure` command (from the subfigure package) defines the group of side-by-side graphics as a single figure and defines each graphics as a subfigure. For example

```
\begin{figure}
   \centering
   \subfigure[Small Box with a Long Caption]{
        \label{fig:subfig:a}          %% label for first subfigure
        \includegraphics[width=1.0in]{box.eps}}
   \hspace{1in}
   \subfigure[Big Box]{
        \label{fig:subfig:b}          %% label for second subfigure
        \includegraphics[width=1.5in]{box.eps}}
   \caption{Two Subfigures}
   \label{fig:subfig}                 %% label for entire figure
\end{figure}
```

produces Figure 20. The label `{fig:subfig:a}` refers to subfigure 20(a), the label `{fig:subfig:b}` refers to subfigure 20(b), and the label `{fig:subfig}` refers to to Figure 20.

(a) Small Box with a Long Caption

(b) Big Box

Figure 20: Two Subfigures

13.3.1 Subfigures Inside Minipage Environments

Like other side-by-side graphics, subfigures are often put inside `minipage` environments. For example

```
\begin{figure}
  \centering
  \begin{minipage}[b]{0.5\textwidth}
     \centering
     \subfigure[Small Box with a Long Caption]{
          \label{fig:subfig:mini:a}   %% label for first subfigure
          \includegraphics[width=1.0in]{box.eps}}
  \end{minipage}%
  \begin{minipage}[b]{0.5\textwidth}
     \centering
     \subfigure[Big Box]{
          \label{fig:subfig:mini:b}   %% label for second subfigure
          \includegraphics[width=1.5in]{box.eps}}
  \end{minipage}
  \caption{Subfigures Inside Minipages}
  \label{fig:subfig:mini}       %% label for entire figure
\end{figure}
```

produces Figure 21 which contains subfigures 21(a) and 21(b).

13.3.2 Minipage Environments Inside Subfigures

Since Subfigure 21(a) consists of only the `\includegraphics` command, the caption in subfigure 21(a) is only as wide as the included graphic. If the subfigure instead consists of the entire minipage, the caption is made as wide as the minipage. For example

(a) Small Box with a Long Caption

(b) Big Box

Figure 21: Subfigures Inside Minipages

```
\begin{figure}
    \subfigure[Small Box with a Long Caption]{
    \label{fig:mini:subfig:a}   %% label for first subfigure
    \begin{minipage}[b]{0.5\textwidth}
       \centering \includegraphics[width=1in]{box.eps}
    \end{minipage}}%
    \subfigure[Big Box]{
    \label{fig:mini:subfig:b}   %% label for second subfigure
    \begin{minipage}[b]{0.5\textwidth}
       \centering \includegraphics[width=1.5in]{box.eps}
    \end{minipage}}
    \caption{Minipages Inside Subfigures}
    \label{fig:mini:subfig}     %% label for entire figure
\end{figure}
```

produces Figure 22. Note that the caption of subfigure 22(a) is considerably wider than that of subfigure 21(a).

(a) Small Box with a Long Caption

(b) Big Box

Figure 22: Minipages Inside Subfigures

13.3.3 Changing Subfigure Numbering

The subfigure labels have two forms

1. One which appears under the subfigure as part of the caption. This is produced by the `\@thesubfigure` command.
2. One which appears when the `\ref` command is used. This is produced by concatenating the output of `\p@subfigure` to the output `\thesubfigure`.

These commands use the `subfigure` counter and the `\thefigure` command, making the subfigure label formating be controlled by the following commands

- The command `\thefigure` prints the current figure number.
- The counter `subfigure` counts the subfigures. The command `\alph{subfigure}` prints the value of the `subfigure` counter in lowercase letters. The command `\roman{subfigure}` prints the value of the `subfigure` counter in lowercase Roman numerals. (see [4, page 98] or [5, page 446] for a list of counter output commands).
- The command `\thesubfigure` by default is `(\alph{subfigure})` which produces (a), (b), etc.
- The command `\@thesubfigure` by default is `\thesubfigure\space` which adds a space between the caption label and the caption.
- The command `\p@subfigure` by default is `\thefigure`

These commands make the default caption labels (a), (b), etc. and the default `\ref` labels 12(a), 12(b), etc. See [10] for controlling the size and font of the subfigure labels.

Subfigure Examples

1. To make the caption labels (i), (ii), etc. and make the `\ref` labels 12i, 12ii, etc. enter the following commands (preferably in the LaTeX file's preamble)

```
\renewcommand{\thesubfigure}{\roman{subfigure}}
\makeatletter
  \renewcommand{\@thesubfigure}{(\thesubfigure)\space}
  \renewcommand{\p@subfigure}{\thefigure}
\makeatother
```

   The `\makeatletter` and `\makeatother` commands protect the `@` signs in the `\renewcommand` statements.
2. To make the caption labels 12.1:, 12.2:, etc. and make the `\ref` labels 12.1, 12.2, etc. enter the following commands

```
\renewcommand{\thesubfigure}{\thefigure.\arabic{subfigure}}
\makeatletter
  \renewcommand{\@thesubfigure}{\thesubfigure:\space}
  \renewcommand{\p@subfigure}{}
\makeatother
```

13.3.4 Adding Subfigures to List of Figures

By default, the List of Figures generated by the `\listoffigures` command includes only figures, *not* subfigures. To add the subfigures the List of Figures, type

```
\setcounter{lofdepth}{2}
```

before the `\listoffigures` command.

14 Minipage Placement Option Details

The manner in which `minipage` environments are vertically aligned may be confusing. For example, one might think the commands

```
\begin{figure}
   \centering
   \begin{minipage}[b]{.25\textwidth}
      \centering \includegraphics[width=1in]{box.eps}
   \end{minipage}
   \begin{minipage}[b]{.25\textwidth}
      \centering \includegraphics[width=1in,angle=-90]{box.eps}
   \end{minipage}
   \caption{\texttt{minipage} with \texttt{[b]} option}
\end{figure}
```

which use the minipage `[b]` options would align the bottoms of the graphics. Instead they produce Figure 23.

Figure 23: `minipage` with `[b]` option

Similarly, one might think the commands

```
\begin{figure}
   \centering
   \begin{minipage}[t]{.25\textwidth}
      \centering \includegraphics[width=1in]{box.eps}
   \end{minipage}
   \begin{minipage}[t]{.25\textwidth}
      \centering \includegraphics[width=1in,angle=-90]{box.eps}
   \end{minipage}
   \caption{\texttt{minipage} with \texttt{[t]} option}
\end{figure}
```

which use the minipage `[t]` options would align the tops of the graphics. Instead they produce a figure which is *exactly* the same as Figure 23.

The `[b]` and `[t]` options produce the same figure because the `minipage` environment's `[b]` option does *not* align the bottoms of the minipages. Rather, it aligns the baselines of the minipages' bottom lines. Similarly, the `[t]` option aligns the baselines of the minipages' top lines. Since the minipages in the above examples only have one line, the `[t]` and `[b]` use the same line for alignment. In this case, the reference point of the minipage is the reference point (original lower-left corner) of the EPS graphic.

14.1 Aligning the Bottoms of Minipages

One method for aligning the bottoms of minipages is to make the bottom of the minipage be the baseline of the minipage. If a line with zero height and zero depth is added inside the minipage after the graphics then the `[b]` option makes the bottom of the minipage be minipage's baseline. The command `\par\vspace{0pt}` creates such a zero-height, zero-depth line. Since the baseline of this zero-depth line is the bottom of the minipage, the `[b]` option now aligns the bottom of the minipage. For example

```
\begin{figure}
   \centering
   \begin{minipage}[b]{.25\textwidth}
      \centering \includegraphics[width=1in]{box.eps}
      \par\vspace{0pt}
   \end{minipage}
   \begin{minipage}[b]{.25\textwidth}
      \centering \includegraphics[width=1in,angle=-90]{box.eps}
      \par\vspace{0pt}
   \end{minipage}
   \caption{Minipages with Bottoms Aligned}
\end{figure}
```

produces Figure 24.

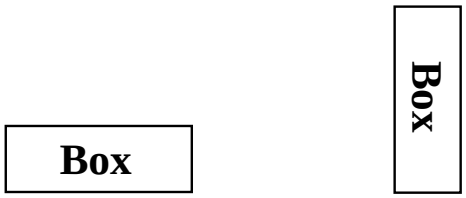


Figure 24: Minipages with Bottoms Aligned

14.2 Aligning the Tops of Minipages

To align the tops of the minipages, one must add a zero-height, zero-depth line to the top of the minipage. Then the `[t]` option makes the top of the minipage be the baseline of the minipage. Preceding `\includegraphics` command by `\vspace{0pt}` inserts a zero-height, zero-depth line above the graphic. Since the baseline of this zero-height line is the top of the minipage, the `[t]` option now aligns the top of the minipage. For example

```
\begin{figure}
   \centering
   \begin{minipage}[t]{.25\textwidth}
      \vspace{0pt}
      \centering \includegraphics[width=1in]{box.eps}
   \end{minipage}
   \begin{minipage}[t]{.25\textwidth}
      \vspace{0pt}
      \centering \includegraphics[width=1in,angle=-90]{box.eps}
   \end{minipage}
   \caption{Minipages with Tops Aligned}
\end{figure}
```

produces Figure 25.

This aligns the tops of the minipages with the current baseline. If it is instead desired to align the tops of the minipages with the top of the current line of text, replace `\vspace{0pt}` with `\vspace{-\baselineskip}`. This topic is mentioned in [5, pages 456-457].

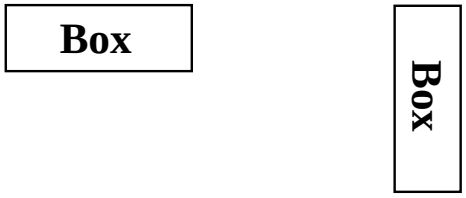


Figure 25: Minipages with Tops Aligned

15 Boxed Figures

The term *Boxed Figure* usually refers to one of two situations

- A box surrounds the figure's graphic but not the figure's caption.
- A box surrounds the figure's graphic and its caption.

The basic method for boxing an item is to simply place the item inside an `\fbox` command, which surrounds the object with a rectangular box. The fancybox package provides boxes of different styles.

15.1 Box Around Graphic

Placing an `\fbox` command around the `\includegraphics` command produces a box around the included graphic. For example, the commands

```
\begin{figure}
   \centering
   \fbox{\includegraphics[totalheight=2in]{file.eps}}
   \caption{Box Around Graphic, But Not Around Caption}
   \label{fig:boxed_graphic}
\end{figure}
```

place a box around the included figure, as shown in Figure 26.

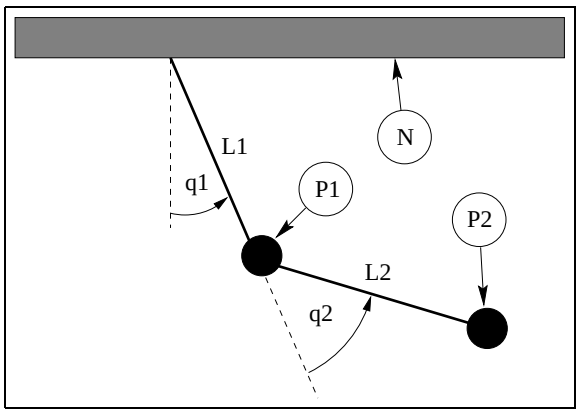


Figure 26: Box Around Graphic, But Not Around Caption

15.2 Box Around Figure and Caption

To include both the figure's graphic and its caption, one may be tempted to move the `\caption` command inside the `\fbox` command. However, this does not work because `\caption` can only be used in paragraph mode, while the contents of an `\fbox` command are processed in LR mode. (LaTeX uses three modes: LR mode, paragraph mode, and math mode. See [4, pages 36,103-5] for an explanation.)

Since the contents of minipage environments and `\parbox` commands are processed in paragraph mode, the `\caption` command can be included in the `\fbox` by enclosing the `\fbox` contents inside a minipage environment or a `\parbox` command. Since both minipages and parboxes require a width specification, there is no direct way to make the `\fbox` exactly as wide the graphic and caption.

For example, the commands

```
\begin{figure}
   \centering
   \fbox{ \begin{minipage}{4 in}
          \centering
```

```
            \includegraphics[totalheight=2in]{pend.eps}
            \caption{Box Around Figure Graphic and Caption}
            \label{fig:boxed_figure}
     \end{minipage} }
  \end{figure}
```

place a box around the figure's graphic and caption, as shown in Figure 27

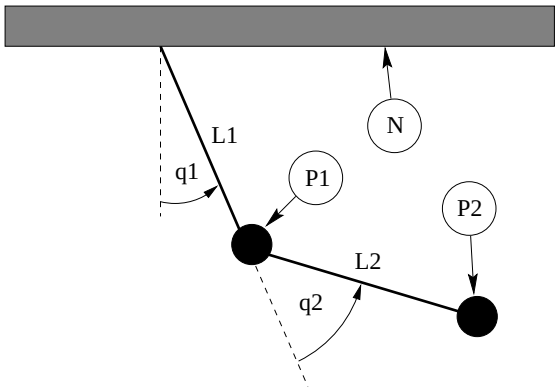


Figure 27: Box Around Figure Graphic and Caption

The determination of a proper minipage width is usually a trial-and-error process. If the caption is wider than the graphic, the minipage can be made as wide as the caption by estimating the caption width with a `\settowidth` command

```
 \begin{figure}
   \centering
   \newlength{\mylength}
   \settowidth{\mylength}{Figure XX:  Box Around Figure Graphic and Caption}
   \fbox{ \begin{minipage}{\mylength}
          \centering
          \includegraphics[totalheight=2in]{pend.eps}
          \caption{Box Around Figure Graphic and Caption}
          \label{fig:boxed_figure_length}
   \end{minipage} }
 \end{figure}
```

15.3 Customizing fbox Parameters

In Figures 26 and 27, the box is constructed of 0.4 pt thick lines with a 3 pt space between the box and the graphic. These two dimensions can be customized by setting the LaTeX length variables `\fboxrule` and `\fboxsep`, respectively, with the `\setlength` command. For example, the commands

```
    \begin{figure}
       \centering
       \setlength{\fboxrule}{3pt}
       \setlength{\fboxsep}{1cm}
       \fbox{\includegraphics[totalheight=2in]{pend.eps}}
       \caption{Graphic with Customized Box}
       \label{fig:boxed_custom}
    \end{figure}
```

place a box with 3 pt thick lines which is separated from the graphic by 1 centimeter, as shown in Figure 28

15.4 The Fancybox Package

In Figures 26, 27, and 28, the `\fbox` command was used to place standard rectangular boxes around the figures. The fancybox package provides four commands `\shadowbox`, `\doublebox`, `\ovalbox`, and `\Ovalbox` which produce other types of boxes.

Like `\fbox`, the separation between these boxes and their contents is controlled by the LaTeX length `\fboxsep`. The length `\shadowsize` is set with the `\setlength` command, as was done for `\fboxrule` and `\fboxsep` in section 15.3. The lines for `\ovalbox` and `\Ovalbox` have thicknesses corresponding to the picture environment's `\thickline` and `\thinline`, which are *not* lengths and thus cannot be changed with the `\setlength` command. The values of `\thickline` and `\thinline` depend on the size and style of the current font. Typical values are 0.8 pt for `\thickline` and 0.4 pt for `\thinline`.

For example, the commands

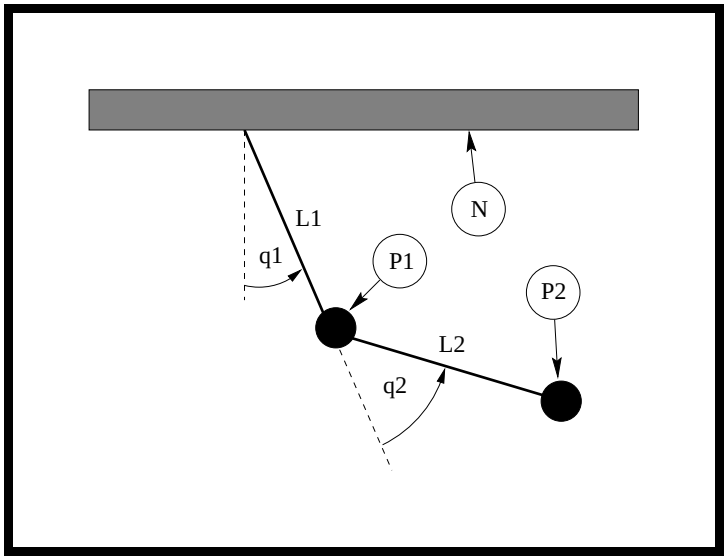


Figure 28: Graphic with Customized Box

Command	Parameters
`\shadowbox{Example}` **Example**	The frame thickness is `\fboxrule`. The shadow thickness is `\shadowsize` (which defaults to 4 pt).
`\doublebox{Example}` **Example**	The inner frame thickness is .75`\fboxrule` and the outer frame thickness is 1.5`\fboxrule`. The spacing between the frames is 1.5`\fboxrule` + 0.5pt.
`\ovalbox{Example}` **Example**	The frame thickness is `\thinlines`. Entering `\cornersize{x}` makes the diameter of the corners `x` times the minimum of the width and the height. The default is `\cornersize{0.5}`. The corner diameter can be set directly by `\cornersize*` command. For example, `\cornersize*{1cm}` makes the corner diameters 1 cm.
`\Ovalbox{Example}` **Example**	`Ovalbox` is exactly the same as `ovalbox` except that the line thickness is controlled by `\thicklines`.

Table 6: FancyBox Commands

```
\begin{figure}
    \centering
    \shadowbox{ \begin{minipage}{3.5 in}
        \centering
        \includegraphics[totalheight=2in]{pend.eps}
        \caption{Shadowbox Around Entire Figure}
        \label{fig:boxed_fancy}
    \end{minipage} }
\end{figure}
```

place a shadow box around the figure's graphic and caption, as shown in Figure 29.

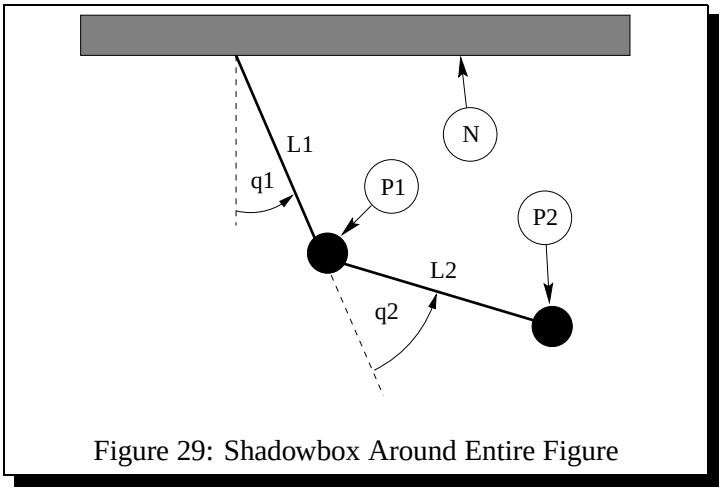


Figure 29: Shadowbox Around Entire Figure

16 Customizing Captions

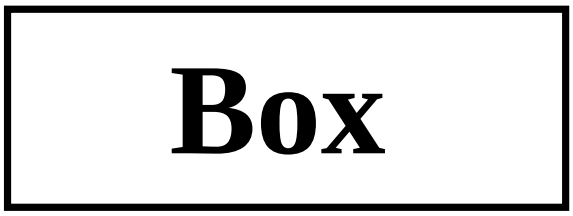
16.1 Captions Next to Figures

The `\caption` command places the caption under the figure or table. Minipage environments can be used to trick the caption command into placing the caption next to the figure. For example, the commands

```
\begin{figure}
   \centering
   \begin{minipage}[c]{3in}
      \centering
      \caption{Caption on the Side}
      \label{fig:side:caption}
  \end{minipage}
  \hfill
  \begin{minipage}[c]{3in}
      \centering
      \includegraphics[width=\textwidth]{box.eps}
  \end{minipage}
\end{figure}
```

produces Figure 30. Likewise, the caption can be placed to the right of the figure by changing the order of the minipages.

Figure 30: Caption on the Side



Since the figure environment defined by the oat package places the caption *below* the body, Figure 30 cannot be produced with the oat package's figure environment. Other aspects of the oat package can be used as long as the `\restylefloat{figure}` command is not issued.

16.2 Controlling Caption Width

Since placing the `\caption` command inside a minipage environment makes the caption as wide as the minipage, this can be used to control the caption width. For example, the commands

```
\begin{figure}
  \centering
  \includegraphics[width=2in]{box.eps}
  \caption{Graphic with a Very, Very, Very, Very, Very, Very Long Caption}
\end{figure}
```

produce the graphic in Figure 31.

Note that the caption in Figure 31 is as wide as the page text. The width of the caption can be limited by placing it inside a minipage environment. For example, the commands

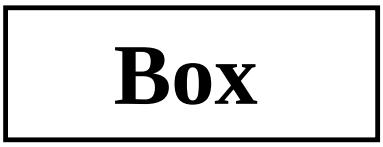


Figure 31: Graphic with a Very, Very, Very, Very, Very, Very Long Caption

```
\begin{figure}
\centering
\begin{minipage}{3in}
  \centering
  \includegraphics[width=2in]{box.eps}
  \caption{Graphic with a Very, Very, Very, Very, Very, Very Long Caption}
\end{minipage}
\end{figure}
```

produces the graphic in Figure 32. The minipage limits the width of the caption in Figure 32 to 3 inches.

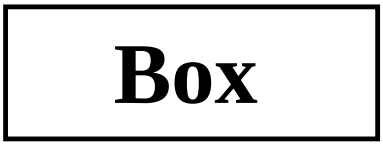


Figure 32: Graphic with a Very, Very, Very, Very, Very, Very Long Caption

A more-general approach to controlling caption width is provided by the caption package and is described in section 16.3.5.

16.3 Caption Package

Since the format of LaTeX figure and table captions (especially for multi-line captions) may not be exactly what users desire, the caption package was written by Harald Axel Sommerfeldt to add flexibility to the caption formatting. Since the original caption package had some bad side-effects (particularly the requirement that it be loaded *after* other packages) it was totally re-written and renamed caption2. Although the caption2 is technically still a beta version, it is quite stable and performs well.

The caption2 package can be used with many types of floats as it officially supports the oat, longtable, and sub gure packages and it also works with the oat g, rotating, supertabular, and wrap g packages.

Reference [12] describes the commands for the original caption package, while the caption2 reference [13] currently includes only minimal documentation. The `test2.tex` test file demonstrates many of the caption2 capabilities.

Syntax: `\usepackage[options]{caption2}`

Where the options are described in Table 7.

16.3.1 Caption Styles

The caption2 package defines the following caption styles

normal Full lines are justified (aligned with both left and right margins) with the last line being left-justified.

center All lines of the caption are centered.

flushleft All lines of the caption are left-justified, leaving the right side ragged.

flushright All lines of the caption are right-justified, leaving the left side ragged.

centerlast All the lines are justified with the last line being centered.

indent Same as "normal" style except that the second and subsequent lines are indented by the length `\captionindent`. Since `\captionindent` is zero by default, a command such as `\setlength{\captionindent}{1cm}` must be used to set the indentation.

hang Same as "normal" style except that the second and subsequent lines are indented by the width of the caption label (e.g., "Figure 12:").

Usually these styles are specified as `\usepackage` options such as

Caption Style	`normal, center, flushleft, flushright, centerlast, hang, indent`	Selects the caption style (see section 16.3.1).
Caption Fontsize	`scriptsize, footnotesize, small, normalsize, large, Large`	Select the fontsize for the caption label (e.g., "Figure 12:") and the caption text.
Caption Label Font Shape	`up, it, sl, sc`	Makes the caption label (e.g., "Figure 12:") have upright, italic, slanted, or small caps shape, respectively. Does not affect caption text.
Caption Label Font Series	`md, bf`	Makes the caption label (e.g., "Figure 12:") have a medium or boldface series font, respectively. Does not affect caption text.
Caption Label Font Family	`rm, sf, tt`	Makes the caption label (e.g., "Figure 12:") have roman, sans serif, or typewriter font, respectively. Does not affect caption text.
One-Line Caption Formatting	`oneline, nooneline`	Controls the formatting for one-line captions (see section 16.3.3)

Table 7: `caption2` Options

```
\usepackage[centerlast]{caption2}
```

which makes all the captions in the document have `centerlast` style. Examples of the caption styles are shown in Figures 33-39.

16.3.2 Changing the Caption Style

The `\captionstyle` command changes the caption style. Placing the `\captionstyle` command inside an environment changes only those captions in that environment. For example, the commands

```
\begin{figure}
   \captionstyle{centerlast}
   \centering \includegraphics[width=3in]{box.eps}
   \caption{Centerlast Caption Style.  Centerlast Caption Style.}
\end{figure}
```

give only the current figure a `centerlast` style because `\captionstyle` is inside the figure environment. The commands

```
\captionstyle{centerlast}
\begin{figure}
   \centering \includegraphics[width=3in]{box.eps}
   \caption{Centerlast Caption Style.  Centerlast Caption Style.}
\end{figure}
```

give subsequent figures a `centerlast` style because `\captionstyle` is outside the figure environment.

16.3.3 One-Line Captions

If the caption is only one line, all of the above styles center the caption. To force the styles to be enforced even for one-line captions, one must include `nooneline` option

```
\usepackage[nooneline,flushleft]{caption2}
```

This formats *all* captions (including one-line captions) with the `flushleft` style. To change the `nooneline` option inside the document, `\onelinecaptionstrue` centers one-line captions while `\onelinecaptionsfalse` formats one-line captions. For example, the commands

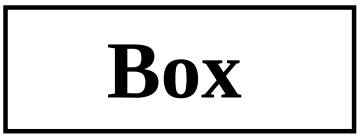


Figure 33: Normal Caption Style. Normal Caption Style.

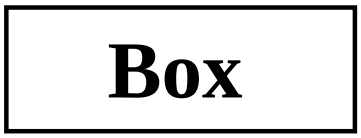


Figure 34: Center Caption Style. Center Caption Style.

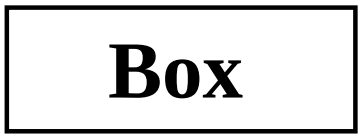


Figure 35: Centerlast Caption Style. Centerlast Caption Style.

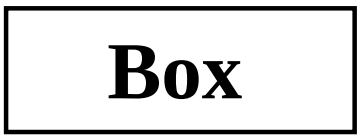


Figure 36: Flushleft Caption Style. Flushleft Caption Style.

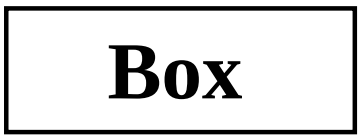


Figure 37: Flushright Caption Style. Flushright Caption Style.

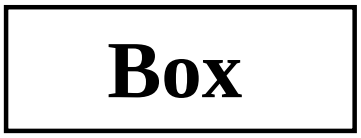


Figure 38: Indent Caption Style. Indent Caption Style.

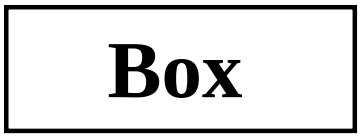


Figure 39: Hang Caption Style. Hang Caption Style.

```
\begin{figure}
   \captionstyle{flushleft}
   \onelinecaptionstrue
   \centering
   \begin{minipage}[c]{2.5in}
      \includegraphics[width=\textwidth]{box.eps}
      \caption{First Caption}
   \end{minipage}
\end{figure}
```

center one-line captions as shown in Figure 40.

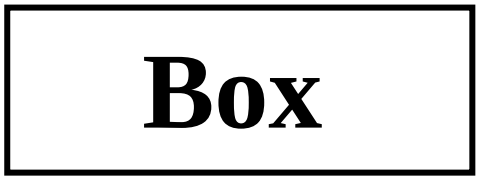


Figure 40: First Caption

The commands

```
\begin{figure}
   \captionstyle{flushleft}
   \onelinecaptionsfalse
   \centering
   \begin{minipage}[c]{2.5in}
      \includegraphics[width=\textwidth]{box.eps}
      \caption{Second Caption}
   \end{minipage}
\end{figure}
```

format one-line captions as shown in Figure 41

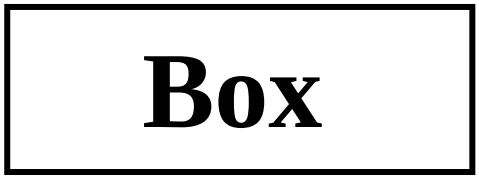


Figure 41: Second Caption

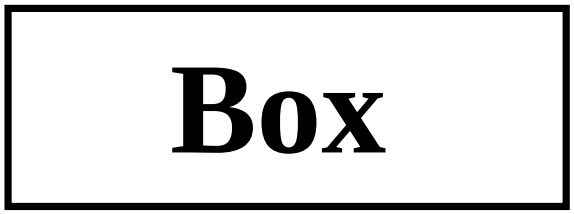


Figure 42: First Line of Caption
Second Line of Caption

16.3.4 Linebreaks in Captions

When the caption fits in one line, it is processed in an `hbox`, which ignores any `\\` or `\par` . Thus one cannot generally specify linebreaks in captions. However, the caption2 package provides the `\onelinecaptionsfalse` command (or `nooneline` option) to turn off this behavior. For example, the commands

```
\begin{figure}[!ht]
   \centering
   \includegraphics[width=3in]{box.eps}
   \captionstyle{center}
   \onelinecaptionsfalse
   \caption{First Line of Caption \protect\\ Second Line of Caption}
   \label{fig:caption:linebreak}
\end{figure}
```

produces the caption in Figure 42 Since `\\` is fragile, it must be preceeded by `\protect`.

16.3.5 Caption Widths

Section 16.2 demonstrated that a `\caption` command appearing in outer paragraph mode can become as wide as the page text as shown in Figure 31. Placing a `\caption` command in a minipage limits the width of the caption to the width of the minipage as shown in Figure 32. The caption2 package provides functions which directly specify the captions' width/margins.

- `\setcaptionwidth{width}` sets the width of the caption to `width`, where `width` can be in any valid TEX units.
- `\setcaptionmargin{mar}` sets the margins to `mar`, making the caption width be the standard width minus 2 times `mar`.
  If `mar` is negative, the caption is made wider than the standard width, which is useful in subfigures and minipage environments.

For example, the commands

```
\begin{figure}
   \setcaptionwidth{3in}
   \centering
   \includegraphics[width=2in]{box.eps}
   \caption{Figure Caption Limited to Three Inches}
\end{figure}
```

make the caption 3 inches wide, as shown in Figure 43.

While the previous example directly set the width of the caption, alternatively the width can be indirectly set by specifying the caption's margin. For example, the commands

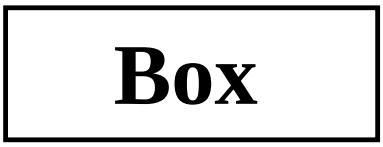


Figure 43: Figure Caption Limited to Three Inches

```
\begin{figure}
   \captionstyle{normal}
   \setcaptionmargin{2in}
   \centering
   \includegraphics[width=2in]{box.eps}
   \caption{Figure Caption With Two-Inch Margins on Each Side}
\end{figure}
```

indent both sides of the caption two inches from the page margins, as shown in Figure 44.

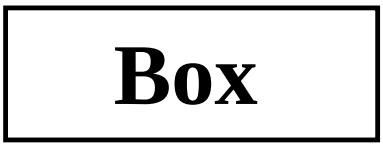


Figure 44: Figure Caption With Two-Inch Margins on Each Side

16.3.6 Caption Font and Delimiter

While the `scriptsize,...,Large` options for `\usepackage{caption2}` change the size of both the caption label (e.g., "Figure 12:") and the caption text, the `up, it, sl, sc, md, bf, rm, sf, tt` options affect only the caption label.

Users can achieve more flexibility by redefining the `\captionfont` and `\captionlabelfont` commands. The caption is created by the following commands

```
{\captionfont%
    {\captionlabelfont \captionlabel \captionlabeldelim}%
    \captiontext}
```

where the `\captionlabel` command produces "Figure 12", the `\captionlabeldelim` command produces ":", and the `\captiontext` command produces the caption text. Thus `\captionfont` affects both the caption label and caption text, while `\captionlabelfont` affects only the caption label.

LaTeX fonts are described by size and three type style components: shape, series, and family ([4, pages 37,115], [5, pages 170-71]). All four of these characteristics can be specified in the `\captionfont` and `\captionlabelfont` commands. For example, the commands

```
\begin{figure}
   \renewcommand{\captionfont}{\Large \bfseries \sffamily}
   \renewcommand{\captionlabelfont}{}
   \centering
   \includegraphics[width=2in]{box.eps}
   \caption{Test Caption}
\end{figure}
```

produce Figure 45. In this example, the `\captionlabelfont` command does nothing. This means that it does not overwrite any font characteristics and all the `\captionfont` settings are carried over to the caption label. Since no shape declaration was specified, the entire caption has the default upright shape.

The commands

```
\begin{figure}
   \captionstyle{normal}
   \renewcommand{\captionfont}{\Large \bfseries \sffamily}
```

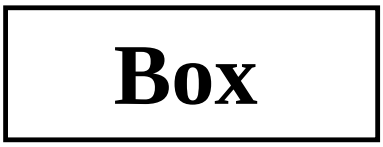


Figure 45: Test Caption

```
   \renewcommand{\captionlabelfont}{\small}
   \centering
   \includegraphics[width=2in]{box.eps}
   \caption{Test Caption}
\end{figure}
```

produce Figure 46. In this example, the `\small` font size in `\captionlabelfont` overwrites the `\Large` font size from `\captionfont`. However, since `\captionlabelfont` does not contain any series or family declarations, the `\bfseries` and `\sffamily` declarations carry over to the caption label.

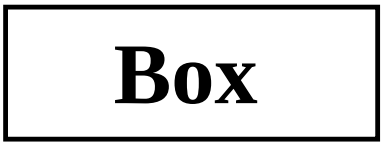


Figure 46: Test Caption

The default colon delimiter can be changed by redefining the `\captionlabeldelim` function. For example, the commands

```
\begin{figure}
   \captionstyle{normal}
   \renewcommand{\captionlabeldelim}{.\quad}
   \centering
   \includegraphics[width=2in]{box.eps}
   \caption{Caption with New Delimiter}
\end{figure}
```

change the delimiter in Figure 47 from the default colon to a period followed by a quad space.

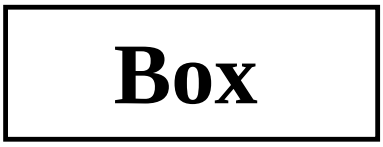


Figure 47. Caption with New Delimiter

16.3.7 Custom Caption Styles

The caption2 package also allows users to create their own caption styles. For example, the following commands

```
\newcaptionstyle{mystyle}{%
  \usecaptionmargin\captionfont%
  {{\centering\bfseries\captionlabelfont\captionlabel\par}%
  \centering\captiontext\par}}

\begin{figure}
   \captionstyle{mystyle}
   \centering
   \includegraphics[width=2in]{box.eps}
   \caption{Customized Caption Style}
\end{figure}
```

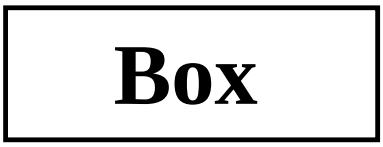


Figure 48
Customized Caption Style

makes the caption label boldface and places it on a separate line from the caption text, as shown in Figure 48.

See the caption2 test file [14] for more user-defined caption style examples.

Acknowledgements

I would like to thank the contributors to the `comp.text.tex` newsgroup, whose posts and replies provided me with the information for this document. In particular, David Carlisle provided a great deal of assistance. I would also like to acknowledge Jim Hafner for providing the procedure in section 10.2. Finally, I would like to thank the readers of previous versions who provided me with feedback.

References

[1] D. P. Carlisle, *Packages in the 'graphics' bundle*, Available from CTAN as `grfguide.tex` or `grfguide.ps`
[2] D. P. Carlisle and S. P. Q. Rahtz, *The graphics package*, Available from CTAN as `graphics.dtx`
[3] D. P. Carlisle and S. P. Q. Rahtz, *The graphicx package*, Available from CTAN as `graphicx.dtx`
[4] Leslie Lamport, *LaTeX: A Document Preparation System*, Addison-Wesley, Reading, Massachusetts, second edition, 1994, ISBN 0-201-52983-1
[5] Michel Goossens, Frank Mittelbach and Alexander Samarin, *The LaTeX Companion*, Addison-Wesley, Reading, Massachusetts, 1994, ISBN 0-201-54199-8
[6] Helmut Kopka and Patrick Daly, *A Guide to LaTeX2ε* , Addison-Wesley, Reading, Massachusetts, 1995, ISBN 0-201-42777-X
[7] Craig Barratt and Michael C. Grant, *The PSfrag system*, Available from CTAN as `pfgguide.tex`
[8] Piet van Oostrum, *Page layout in LaTeX*, Available from CTAN as `fancyhdr.tex`
[9] Leonor Barroca, *The rotating package*, Available from CTAN as `rotating.dtx`
[10] Steven Douglas Cochran, *The subfigure package*, Available from CTAN as `subfigure.dtx`
[11] Timothy Van Zandt, *Documentation for fancybox.sty*, Available from CTAN as `fancybox.doc`
[12] Harald Axel Sommerfeldt, *The caption package*, Available from CTAN as `caption.dtx`
[13] Harald Axel Sommerfeldt, *The caption package*, Available from CTAN as `caption2.dtx`
[14] Harald Axel Sommerfeldt, *Test of the caption package*, Available from CTAN as `test2.tex`

StarTEX—a TEX for beginners

Dag Langmyhr

Department of Informatics
University of Oslo
Norway
dag@ifi.uio.no

Abstract

This article describes StarTEX, a new TEX format for students writing their first report and other novice users. Its aim is to provide a simpler and more robust tool for users with no previous knowledge of TEX and LATEX.

1 The problem

Students taking courses at our department are required to write short project reports, and LATEX [2] has been the preferred tool. Several years experience has, however, shown us that LATEX is not ideal for this.

This project report is the first encounter most students have with LATEX, and they face many problems:

- The major problem is the error messages. They are very terse at best, and since they are sometimes produced by LATEX and at other times by TEX, understanding the messages requires reasonably good knowledge of both systems. Most students tend to look only at the line numbers when examining their error logs.
- LATEX is not very robust; trivial syntax errors can cause a serious burst of confusing error messages, like when you forget a `\\` prior to `\hline` in an `array` environment.
  You can also experience undesired effects if you use the commands incorrectly, for instance if you write
  `\abstract{`*text*`}`
  rather than the correct
  ```
  \begin{abstract}
    text
  \end{abstract}
  ```
  This error produces no error message, but will cause the whole article to be set in a smaller font.
- LATEX does not hide the primitive commands of TEX, making it possible for the users to access them accidentally. For example, one of our users defined a macro for her name:
  ```
  \def \else {Else Hansen}
  ```
  This error alone produced more then 100 error messages.
- LATEX uses ten special characters: #, $, %, &, ~, ^, _, {, } and \. Users need to remember that these characters are special, and they must learn which commands are necessary to produce them if they are required in the text. Fewer special characters would be an advantage.
- The command notation `\xxx` used in LATEX often causes problems with the space following it.
- LATEX has borrowed its error recovery philosophy from plain TEX: the user is expected to manually correct each detected error to allow LATEX to proceed. The problem with this approach is that you will get many confusing error messages if you do not correct the error properly.
  None of our students use this interactive recovery facility; they either restart after having discovered the first error, or they let the processing run to completion without any interaction. An automatic error recovery scheme like that employed by compilers would be a great benefit for these users.
- LATEX provides a mixture of structural mark-up commands as well as visual mark-up. The advantage is that experienced users can achieve the visual appearance they desire; the disadvantage is that less experienced users—particularly those who have used other document processing tools—spend too much of their time trying to coerce LATEX into producing exactly the layout they think is proper.
- LATEX is a large system and running it is not as fast as for instance plain TEX. For instance, a $2\frac{1}{2}$ page sample document takes from 2.8 seconds on a Sun SparcStation20 to 8.7 seconds on a Silicon Graphics Indy. Since novice users tend to process their documents very frequently to remove errors or test the effect of a feature, execution times do matter—even in this range.

2 The requirements

All these problems indicate that LATEX in its present form is not the tool we want for our students, at least not for their first report. We want a document processing program with the following properties:

- It must be based on TEX to achieve the desired quality in mathematical formulae.

- It should use a different notation for its mark-up commands; one which caused less confusion concerning spaces and has fewer special characters.
- It must hide all the internal TEX commands; this is the only safe way to avoid students using them accidentally.
- It must be small and easy to understand, so that it may easily be adapted to the particular need of each installations.
- It should contain structural mark-up commands only, and no visual mark-up.
- It should be robust.
- It should produce better error messages. If possible, no messages from TEX should ever appear. If this is impossible, error messages from TEX should be preceded by a message produced by the new tool.
- Since most students tend to just disregard all messages about under- and over-full boxes, it should try to reduce the number of such messages.
- It should run in nonstop mode and use automatic error recovery to detect as many genuine errors as possible.
- It should be as fast as plain TEX.
- The command handling should be insensitive to uppercase and lowercase. This is not an important issue, but case confusion has caused problems for some.

3 The solution

Attempting to achieve the goals mentioned above, StarTEX was designed. The name was chosen to indicate that it was a *Starters' TEX*.

StarTEX is a new format, and is thus a simple cousin of $\mathcal{AMS}$-TEX [5] and LATEX. It is built on top of the plain TEX [1] commands.

4 The notation

At the EuroTEX conference in Arhem in September last year, Philip Taylor [6] proposed a different notation for (LA)TEX commands:

<xxx> rather than \xxx.

I decided to use this notation in StarTEX as it solves many of our problems:

- Spaces following the command are no longer a problem. There is no need for special rules like "When a space comes after a control word, it is ignored by TEX." [1, p. 8].
- Only one special character is needed: <. The characters #, $, %, &, ~, ^, _, {, } and \ can be defined to be just ordinary characters.
- The command name may contain almost any character, not just letters.
- The scheme is easy to implement: all that is required is to make < an active character, and let the corresponding command regard everything up to the following > as a parameter.
- Since all commands are called through this interface, it is easy to make all internal TEX commands invisible.
- It is easy to check whether the user command is defined, and provide suitable error recovery if it is not.
- It is easy to \lowercase the user command, thus making the command handling insensitive to case.
- This command notation is the same as in HTML [4] with which many students are familiar.

I could have used any bracketing symbol pair, like [xxx] or {xxx} or /xxx\, but I chose <xxx> because it resembles HTML and because < and > are not used very frequently.

4.1 Command parameters

A few commands need a parameter to specify non-printing matter like a file name or a label. I chose to use square brackets for this, as in

<ref>[*label*]

Using a special notation indicates more clearly that the parameter is not to be typeset.

5 The command set

The set of available StarTEX commands was chosen with the following aims in mind:

- There should be sufficient commands for writing a student report, but otherwise there should be as few commands as possible.
- There should be no commands for visual mark-up, only structural specifications.
- The commands should have a form that makes them easy to check for errors, and to automatically recover from the errors.

In table 1 are listed most of the StarTEX commands with their LATEX counterpart.

5.1 Paragraph separation

It was decided to use <p> to separate paragraphs, as in HTML. Even though the blank line used by (LA)TEX is easier to type, it does cause problems with indentation of the paragraph following an environment like a list. Using <p> alleviates this problem.

Another advantage of using the <p> notation is that it can be employed as line separator (like \\ in LATEX) in environments where the concept of paragraph makes little sense, as in the <title> or <author> environments. This provides a double benefit: a special command for line breaking is no longer necessary, and using <p> in a <title> environment is now legal.

5.2 Font selection

A few commands for font selection are necessary, but my belief is that <b> (for **bold text**), <i> (for *italic*) and <tt> (for `typewriter text`) form a sufficient set of

	StarTEX	**LATEX**
Document bounds	`<body>`*text*`</body>`	`\begin{document}`*text*`\end{document}`
Document style	`\style[`*style file*`]`	`\documentclass{`*style file*`}`

Document head	`<title>`*text*`</title>`
	`<author>`*text*`</author>`
	`<info>`*text*`</info>`
		`\title{`*text*`}`
		`\author{`*text*`}`
		`\date{`*text*`}`

Font change	`<b>`*text*`</b>`
	`<i>`*text*`</i>`
	`<tt>`*text*`</tt>`
		`\textbf{`*text*`}`
		`\textit{`*text*`}`
		`\texttt{`*text*`}`

Paragraph break	`<p>`	⟨blank line⟩
Mathematical formula	`<math>`*formula*`</math>` `<displaymath>`*formula*`</displaymath>`	`\(`*formula*`\)` `\[`*formula*`\]`

Sectioning	`<h1>`*text*`</h1>`
	`<h2>`*text*`</h2>`
	`<h3>`*text*`</h3>`
	`<h4>`*text*`</h4>`
		`\section{`*text*`}`
		`\subsection{`*text*`}`
		`\subsubsection{`*text*`}`
		`\paragraph{`*text*`}`

Itemized list	`<list>`
	`  <item>`…
	`  :`
	`</list>`
		`\begin{itemize}`
		`  \item`…
		`  :`
		`\end{itemize}`

Enumerated list	`<list>`
	`  <numitem>`…
	`  :`
	`</list>`
		`\begin{enumerate}`
		`  \item`…
		`  :`
		`\end{enumerate}`

Description list	`<list>`
	`  <textitem>`*text*`</textitem>`…
	`  :`
	`</list>`
		`\begin{description}`
		`  \item[`*text*`]`…
		`  :`
		`\end{description}`

PostScript figure	`<psfig>[`*file name*`]`*caption text*
	`</psfig>`
		`\begin{figure}`
		`  \caption{`*caption text*`}`
		`  \begin{center}`
		`  \epsfig{file=`*file name*`,`…`}`
		`  \end{center}`
		`\end{figure}`

Table	`<table>`*caption text*
	`  <row>`*text*`<col>`*text*`<col>`…
	`  <row>`*text*`<col>`*text*`<col>`…
	`  :`
	`</table>`
		`\begin{table}`
		`  \caption{`*caption text*`}`
		`  \begin{center}`
		`  \begin{tabular}{|c|`…`}\hline`
		`  `*text*`&` *text*`&` …`\\ \hline`
		`  `*text*`&` *text*`&` …`\\ \hline`
		`  :`
		`  \end{tabular}`
		`  \end{center}`
		`\end{table}`

Footnote	`<footnote>`*text*`</footnote>`	`\footnote{`*text*`}`
Unformatted text	`<code>`*text*`</code>`	`\begin{verbatim}`*text*`\end{verbatim}`
Cross references	`<label>[`*label*`]` `<ref>[`*label*`]`	`\label{`*label*`}` `\ref{`*label*`}`, `\pageref{`*label*`}`
Comments	`<comment>`*text*`</comment>`	`%`*text*⟨end-of-line⟩
User macro	`<define><`*name*`>`*definition*⟨end-of-line⟩	`\newcommand{\`*name*`}{`*definition*`}`

Table 1: StarTEX command overview

Index	Data
12	199
17	0

Table 2: A small table sample

commands. The commands may of course be nested to provide for instance *italic typewriter text*.

Some might argue that these commands are visual rather than structural, and that the HTML approach of providing a wider selection of structural commands like <dfn> for definitions, for emphasis, <kbd> for keyboard input and <samp> for literal characters, is more logical. My own experience is that there are seldom enough definitions to suit my needs, so I will for instance use a specification like when I really want to indicate a reserved word in a programming language. Providing a few simple type changing commands is simpler.

5.3 PostScript figures

Since nearly all figures used in LATEX documents at our department are PostScript files, it seems reasonable to specialize the interface for this. The notation

<psfig>[*file name*]*caption text*</psfig>

was chosen as only two keywords were necessary. All figures are automatically scaled and they float to the top of the current or following page.

5.4 Tables

The notation for tables was also chosen to be as simple as possible, and to ease error detection and recovery. Only very regular tables are catered for, but this is the price one has to pay for a simple notation.

A table is a complex structure, with entries in columns within rows inside the table, but a notation was found which will seldom give grouping errors:

<table>*caption text*
<row>*text*<col>*text*<col>...
<row>*text*<col>*text*<col>...
:
</table>

Every <row> starts a new row, and each <col> starts another column. The text prior to the first row is regarded as the table caption.

The number of columns is determined automatically. All columns are centered, and a grid of horizontal and vertical rules is always added. For example, the code

```
<table>
  A small table sample
  <row> <b>Index</b> <col> <b>Data</b>
  <row> 12 <col> 199
  <row> 17 <col> 0
</table>
```

will generate the table shown as table 2.

5.5 Document styles

All documents need some adaption to conform to a particular style. I propose to let the user decide this by stating

<style>[*style file*]

The style file is written in plain TEX and contains the necessary definitions and modifications. Since the user has no visual mark-up commands at his or her disposal, all design decisions are made by the style designer. This makes it easier to have all reports conform to the approved standard.

My hope is that each site using StarTEX will develop styles of their own. These styles should be comprehensive, so the user should only have to specify that one style. For instance, our style ifi-report defines

- the page size (A4 paper),
- Norwegian format of <today> and <now>,
- Norwegian translations of fixed texts like “Figure” and “Table”,
- the page headers and footers, and
- various minor typographic details.

5.6 Cross references

StarTEX uses more or less the same mechanisms for cross references as LATEX. Interesting sections, figures and tables are given a label using the <label> command, which may then be referenced using the <ref> command.

The appearance of the reference is defined by the document style, but will normally contain the page number if the reference is a different page; there is thus no need for a \pageref command. (This is similar to the varioref package [3].

5.7 Mathematical formulae

One of the most important reasons for choosing a typesetting system based on TEX is its ability to typeset mathematical formulae. All the math mode commands available in (LA)TEX are implemented in StarTEX, and most of them use a notation similar to HTML version 3.0. For example, the formula

$$\int_1^\infty \frac{f(x)}{1+x}\partial x$$

is typed as

```
<body>
<title> <startex><--->A <tex> for beginners </title>
<author> Dag Langmyhr<p> Department of Informatics<p>
  University of Oslo<p> <tt>dag@ifi.uio.no</tt>
</author>
<info> <today> </info>

<abstract> This document describes <startex>, a special <tex>
  format for students writing their first project report.
</abstract>

<h1> The basic philosophy of <Startex> </h1>
<Startex> was designed for novice <tex> users. It employs a
different notation and a different set of commands from <latex>,
and the idea is that this makes it more user-friendly for these
users than plain <tex> or <latex>.

<p>
The notation used in <startex> resembles HTML and some of the
commands are the same, but the philosophy of the two is
different. HTML was designed to display hypertext information
on a computer screen, while <startex> is used to produce a
student report on paper.
</body>
```

Figure 1: An example StarTEX document

```
<displaymath>
  <int><sub>1</sub><sup><infinity></sup>
    <frac>f(x)<over>1+x</frac>
    <partial>x
</displaymath>
```

5.8 User-defined macros

It was decided to allow the users to define their own commands, but with the following restrictions:

- The macros may not have parameters.
- No macros may be redefined.

The StarTEX notation

was chosen to make error recovery easier. There is now no chance of a runaway definition, like you would get in (LA)TEX if you forgot a final `}`.

5.9 Various other commands

In table 3 are shown the few remaining StarTEX commands.

5.10 An example

In figure 1 is shown an example document using some of the StarTEX commands.

6 Other design decisions

6.1 Error recovery

As mentioned previously, StarTEX can employ the `<xxx>` notation to detect errors and provide some error recovery. For instance, it keeps track of both the current and the outer environments, and which commands should be used to exit those environments. This means that it can detect and remedy the following situations:

- A missing terminator `</xxx>` will be detected when the outer environment is finished. In this case, both environments will be exited, and you would get an error message like

  ```
  ** StarTeX error detected on line 7:
     <i> on line 7 terminated by </b>.
     An extra </i> has been inserted.
  ```

- A superfluous terminator `</xxx>` will be recognized as such, and ignored, and the user would be notified with the following error message:

  ```
  ** StarTeX error detected on line 15:
     <body> on line 1 terminated by </b>.
     The </b> will be ignored.
  ```

6.2 Paragraph parameters

LATEX is a program for quality typesetting, and this is reflected in the standard parameters for paragraph breaking. Even paragraphs that look quite good to an untrained eye may produce messages about under- or over-full boxes. When LATEX is unable to find a set of breaks it regards as acceptable, the result may be truly horrible, with words sticking into the margin, or all excess space put into the first line. This occurs quite often in Norwegian which has many long compound words. An experienced LATEX user will easily detect the problem word and fix that or rephrase the text, but novice users seldom understand these messages and tend to ignore them.

Symbol	StarTEX code
<	<lt>
>	<gt>
–	<-->
—	<--->
⟨a tie⟩	<~>
…	<...>
⟨today's date⟩	<today>
⟨the present time⟩	<now>
TEX	<tex>
LATEX	<latex>
StarTEX	<startex>

Table 3: The remaining StarTEX commands

All the messages about over-full and under-full boxes create another problem for the LATEX novices. Since many of them use tools (like AUC-TEX [7]) that run LATEX in non-stop mode, they get pages and pages of serious error messages intertwined with innocuous warnings, so they tend to just ignore all the messages as long as the printed result looks acceptable to them.

StarTEX sets its standard parameters for very loose typesetting with high values for `\tolerance` and `\emergencystretch`. The reasons for this are:

- If a good set of paragraph breaks exists, TEX will still choose that.
- Since the users tend to ignore messages about bad breaks, it is better to have a loosely broken paragraph than the very bad result you may get when TEX has to give up.
- The results achieved this way are at least as good as those produced by other typesetting and text-processing software.

This solution does not solve the problem of obtaining good paragraph breaks, but experience so far has shown that it goes a long way.

7 Concluding remarks

StarTEX has been completed and is being introduced to the students the coming term. It has—in my opinion–achieved most of the specified goals, but not all.

- It is quite small, consisting of fewer than one thousand lines of TEX code plus documentation. Whether the code is easy to understand is for others to judge.
- It is moderately robust. Most simple errors are handled by StarTEX, but grave ones still confuse it.
- It is reasonably fast; the $2\frac{1}{2}$ page example document mentioned at the beginning of this article is processed in 0.9 and 1.6 seconds, respectively.

Even though the users are taught a different format with a different command syntax, I believe StarTEX will serve as a suitable introduction to LATEX and document processing, because it provides training in the *concepts* of LATEX and structural mark-up.

(An analogy from computer science: The programming language C is widely used, and most programmers should know it. It is, however, a language for experts, so a common view is that students should first learn the concepts of programming in a different language before being exposed to C.)

The invention of StarTEX is not intended as any kind of criticism against LATEX, which is still our main tool for larger documents and for the more experienced users. The aim of StarTEX is to help one specific group of users, and provide them with a gentler introduction into the world of (LA)TEX.

On the other hand, StarTEX can be regarded as a tribute to TEX which so easily allows one to produce a different user interface to its powerful mechanisms.

7.1 Why not use HTML?

Some users have asked why we do not use HTML when the notation is so similar. There are several reasons for that:

- There is no yet final definition of HTML. There are several versions available, in addition to the inventions of various software companies. Nobody knows what HTML will look like a few years from now.
- HTML is growing very complex, with many constructs of little interest to the student writing a report.
- It is difficult to write a robust parser of HTML in TEX.

7.2 Availability

If anyone is interested in obtaining a copy of StarTEX, they can find it available for anonymous FTP on `ftp.ifi.uio.no` in the directory `pub/tex/startex`.

References

[1] Donald E. Knuth. *The TEXbook*. Addison-Wesley, 1991.

[2] Leslie Lamport. *LATEX user's guide and reference manual*. Addison-Wesley, 1994.

[3] Frank Mittelbach. The `varioref` package. Part of the LATEX2ε distribution, May 1995.

[4] Dave Raggett. Hypertext markup language specification version 3.0 draft. Available at `http://www.w3.org/pub/WWW/MarkUp/html3/`, March 1995.

[5] Michael Spivak. *The joy of TEX*. American Mathematical Society, 1986. The guide to $\mathcal{AMS}$-TEX.

[6] Philip Taylor. TEX: an unsuitable language for document markup? Talk given at the EuroTEX 1995 conference; does not appear in the proceedings, 1995.

[7] Kresten Krab Thorup. AUC TEX. An Emacs mode for editing (LA)TEX code; available from `http://www.iesd.auc.dk/~amanda/auctex/`, 1996.

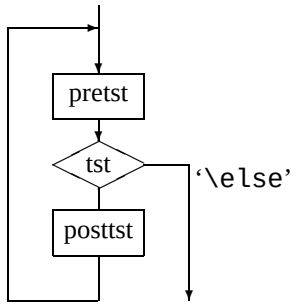
Paradigms: Loops

Kees van der Laan

1 BLUe's Design VII

Hi folks. When you like plain's `\loop`-s so much as I do then this is for you. Hang on Dek's loop implements the flow, The TEXbook 219

```
\def\loop#1\repeat{%
  \def\body{#1}\iterate}
\def\iterate{\body
   \let\next\iterate
   \else\let\next\relax
   \fi\next}
```



with as pseudosyntax for the tags

`\loop`⟨*pretst*⟩`\if...`⟨*posttst*⟩`\repeat`.

Special cases result when either ⟨*pretst*⟩, or ⟨*posttst*⟩ is empty. The former is equivalent to for example PASCAL's **while** ... **do** ..., and the latter to **repeat**... **until**. From this I conclude that all those `\while` casu quo `\repeat` flavors are not needed. Syntactic sugar? Yes, IMHO, with all respect.

In practice we all need repetitions either via a loop or via tail recursion, which by the way are equivalent. The loop notation is simpler to use than tail recursion, I guess.

2 Van der Goot's loop

Van der Goot implemented the loop construction as a straight tail recursion. An eye-opener I have simplified it into the following.[1]

```
\def\Loop#1\Pool{#1\Loop#1\Pool}
\def\Break#1\Pool{}
%a trivial example of use
\count0=10
\Loop a \advance\count0 by-1
   \ifnum\count0=0 \expandafter\Break\fi
\Pool
\end
```

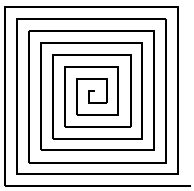
For more details see his Midnight suite, `loop.doc` and `loop.tex`.[2]

Remark. Either `\expandafter` is needed or the `\Break` should take `\fi` as replacement text. The above with `\expandafter` is convenient with nesting of loops.

3 Loops in markup

I used a loop in the markup for turtle graphics, especially for the stochastic Sierpiński carpets—to throw the dice repeatedly—and for a spiral. In the markup for the spiral below `\E`, ...`\N` mean draw in the directions east, ...north.

Example (Spiral)



```
$$\vbox to3cm{\vss
 \hsize0pt \offinterlineskip
 \unitlength.5ex \y0pt \x0pt
 \loop\N{\the\k}\advance\k+1
      \W{\the\k}\advance\k+1
      \S{\the\k}\advance\k+1
      \E{\the\k}\advance\k+1
 \ifnum\k<30 \repeat
\vss}$$
```

Remark. One can also start at a corner. However, for reuse of these kinds of pictures I consider it more convenient to start at the center of symmetry.

Most of the time I use the tail recursion via my FIFO or (binary) tree macros. My favorite FIFO example is the outline exercise The TEXbook ex11.5. I solved this via nested FIFO processing

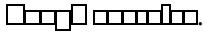
- words via `\fifow`, and
- characters via `\fifo`.

Example (Outlines)

```
\fifow Tough  exercise \wofif \unskip.
%with
\def\fifow#1 {\ifx\wofif#1\wofif\fi
   \processw{#1}\fifow}
\def\processw#1{\fifo#1 \ofif\ }
\def\process#1{\boxit#1}
\def\boxit#1{\setbox0=\hbox{#1}%
  \hbox{\lower\dp0\vbox{\hrule
   \hbox{\vrule\phantom#1\vrule}%
   \hrule}}}
```

[1] The unusual in the coding is the infinite loop. I have to unlearn much. In the old days of ALGOL 60 I was taught to avoid side-effects and infinite loops of course. With the expression language ALGOL 68 all storing was a side-effect. Now with the interpretive languages it is beneficial to think in infinite loops.

[2] Available on the TEX-Alive TUG CD.



Another nice example is writing lines verbatim to a file, that is manmac's `\copytoblankline` recast in FIFO terms.[3]

Example (Sorting citation lists)

Suppose, we have

```
\def\lst{\\\ia\\\ib\\\ic}
\def\ia{314}\def\ib{27}\def\ic{1}
```

then sorting, the invoke of `\lst`, yields 1, 27, 314.

The above is obtained as follows.

```
\def\dblbsl#1{\ifnum#1<\min\let\min=#1\fi}
%
\Loop\ifx\empty\lst\expandafter\break\fi
   \def\\{\let\\=\dblbsl\let\min=} %space
   \lst%find minimum
   \min%typeset minimum
   {\def\\#1{\ifx#1\min \else\nx\\%
    \nx#1\fi}\xdef\lst{\lst}}%
\Pool
%with auxiliaries
\def\Loop#1\Pool{#1\Loop#1\Pool}
\def\break#1\Pool{}
```

The coding implements the looping of the basic steps

- find minimum (via `\lst`, and suitable definition of Dek's active list separator `\\`)
- typeset minimum (via `\min`)
- delete minimum from the list (via another appropriate definition of the list element tag).

Remarks. `\dblbsl` is mnemonics for double backslash. The deletion of elements from the list works with `\ifx`. The variant with `\ifnum` does not work as such because a `\relax` is inserted by TEX.[4]

The problem occurred in sorting citation lists of bibliography references which are specified by their symbolic names.[5]

Example (Reading a file line by line)

```
\Loop\ifeof\readfile\Break\fi
  \read\readfile to\inputline
  <process \inputline>
\Pool
%with auxiliary
\def\Break#1\Pool{\fi}
```

This use occurred in the Convertor Assistant BLUe-2-LATEX, well ... it is the main loop.[6]

3.1 Relevancy

From my experience as exemplified above, I conclude that loops are mainly of interest for macro writers and not so much for the casual (La)TEX user.

4 Macro writers attention

Knuth's loop does not allow for the use of `\else` in the exit code because it is already part of the macro `\loop`. Kabelschacht 1987—and Spivak 1989—needed the use of `\else`.

Example (Kabelschacht's loop)

Kabelschacht removed the `\else` from the loop code as follows.

```
\def\loop#1\repeat{\def\iterate{#1\ea
   \iterate\fi}\iterate}
%a trivial example of use
\count0=10
\loop\advance\count0 by-1
   \ifnum\count0=0
   \else do what has to be done
\repeat
```

Remarks. The exit is via the then-branche in contrast with Knuth's loop. Suppressed are some efficiency aspects with respect to storage. Kabelschacht's claim that his loop is a *generalization* of plain's loop must be seen in the light of not being restricted to quit a loop via the else-branch.

The reason, I can think of, for introducing another loop macro, while the most general form has been implemented already, is the existence of commands like `\ifvoid`, and `\ifeof`, and the absence of their negatives `\ifnonvoid`, respectively `\ifnoneof`. In those cases we like to continue the loop via the `\else` branch. For the latter case this means to continue the loop when the file is *not* ended. This can be obtained via modifying the loop à la Kabelschacht or more elegantly via van der Goot's macro.

Another approach is to use a `\newif` parameter, better known as 'boolean' or 'logical' in other programming languages, together with Knuth's loop. A `\newif` parameter, `\ifneof`, can be used to test for an end of file or casu quo, an end of a list.

```
\ifx\lst\empty\neofalse\else\noetrue\fi\ifneo
```

Example (Reading a list via Knuth's loop)

```
\newif\ifneo
\def\lst{a b c}\neotrue
\loop\ifx\lst\empty\neofalse\fi
   \ifneo \def\lst{}
\repeat
```

Related to the coding of the logical ¬ are the codings of the logical `and`, ∧, and `or`, ∨, as can be seen from the accompanying table.

[3] See FIFO and LIFO sing the BLUes.

[4] The `\relax` can be suppressed by inserting an unexpandable token like `\noexpand \empty`. Courtesy Bernd Raichle.

[5] In BLUe's bibliography I have explained how to circumvent the sorting of citation lists of references.

[6] Line by line is a white lie. A line or a group embraced by curley braces is read.

Functional code	TEX coding
¬\if...	\if...\notfalse\else \nottrue\fi\ifnot
\if...∧\if...	\andtrue\if...\if... \else\andfalse \else\andfalse\fi\fi \ifand
\if...∨\if...	\ortrue \if...\else\if...\else \orfalse\fi\fi \ifor %alternative \orfalse \if...\ortrue\fi \if...\ortrue\fi \ifor

with the \newif-s: \ifnot, \ifand, and \ifor.

5 Nesting of loops

My favorite example of nesting of loops is the bubble sort.[7]

```
\def\bubblesort{%
%Data in defs \1, \2,...\<n>.
{\loop\ifnum1<\n{\k\n
  \loop\ifnum1<\k \advance\k-1
     \cmp{\deref\k}{\deref\n}%
   \ifnum1=\status\xch\k\n\fi
  \repeat}\advance\n-1
 \repeat}}%end \bubblesort
%with auxiliary
\def\deref#1{\csname\the#1\endcsname}
\let\cmp\cmpn %from blue.tex or provide
\def\cmp#1#2{
%Yields status=0, 1, 2 for =, >, <
...}
```

5.1 Inconsistency pitfall

I experience the non-expansion of a counter variable—or the non-dereference of it as you wish—within a \csname counter-intuitive. The counter must be preceded by \the, see \deref. I understand the difference with the situation that a value might be assigned but in this construct this is out of order.

Another nesting of 'loops' is in the earlier mentioned linear sorting of citation lists, where the 'inner loop' is the \xdef.

5.2 Braces around inner loops are mandatory

Pittman argued that there is a need for other loop codings.

> 'Recently, I encountered an application that required a set of nested loops and local-only assignments and definitions. TEX's \loop ... \repeat construction proved to be inadequate because of the requirement that the inner loop be grouped.'

In 'Syntactic Sugar' I have shown that his problem can be solved from a table point of view.[8]

However, Pittman was definitely right with respect to bracing the inner loop, because of the parameter separator \repeat. If braces are omitted, the first \repeat is mistaken for the outer one, with the result that the text of the outer loop will *not* become the first \body. The good way is, to make the inner \repeat invisible for the outer loop level, by enclosing the inner loop in braces.

With non-explicit nesting—for example the inner loop is the replacement text of a macro—we still need scope braces, because otherwise the \body of the outer loop will be silently redefined by the body of the inner loop.

5.3 Dialogue with TEX

Nesting of loops occurs when in dialogue with TEX only certain answers are allowed.

Example (Checking user input)

```
%<TeX_Marker>
%Insist on allowed answers only
\def\iw{\immediate\write0}
\def\readyesornotoanswer{{%
   \def\yes{yes}\def\no{no}
   \Loop\iw{Please provide yes or no:}
      \read-1 to\answer
      \ifx\answer\yes\ea\Break\fi
      \ifx\answer\no \ea\Break\fi
    \Pool\xdef\answer{\answer}}}
%
\let\yes\Break\let\no\relax
\endlinechar-1 %TB20.18
\Loop\message{Are you happy?}
   \readyesornotoanswer
   \ea\csname\answer\endcsname
   \iw{Too bad, I insist...}
\Pool
\bye
```

Remarks. The \xdef makes the answer globally available. En-passant a little 'switch functionality in TEX' is realized via \csname.

6 Hidden counter

I like a hidden counter very much.[9] In TEX we don't have a garbage collector and therefore there is no gains in storage. However, to alleviate the user from the details of the allocation of storage of the counter I provided the following.[10]

```
\def\preloop{%To create loopcnt, a
            %local loopcounter
   \bgroup \advance\count10 by 1
   \countdef\loopcnt=\count10
            %Symbolic name
   \loopcnt=1 %(default)
}%end \preloop
\def\postloop{\loopcnt=0 %Restore
   \egroup}%end \postloop
```

[7] \cmp stands for comparison. \xch stands for exchange. For a pseudocode see Paradigms: Sorting.

[8] It is also in the tables chapter of PWT.

[9] As in ALGOL 68, METAFONT/MetaPost, or PostScript.

[10] Bernd Raichle has implemented local storage allocation macros. Although I can see the benefits of his robust approach especially together with push-the-button packages like LATEX, I refrained from it, because for my applications I know that the bounds—only 256 locations are available—are not severe. Moreover, intelligibility is hampered, and I like to add just a little to TEX, as little as possible.

I used the above in the coding of the Tower of Hanoi game. The paradigm in there is that it is used together with `\aftergroup`—shortcut `\ag`—to create *dynamically* a data structure, i.e. the tower of which the size is specified by the user.

```
%   \def\I{\disksep\i\disksep\ii
%        \disksep\iii...\disksep\‘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\disksep\ea\ag\csname
         \romannumeral\loopcnt\endcsname
    \ifnum\loopcnt<\n
    \advance\loopcnt by1
  \repeat   \ag}
\postloop
```

Note that `\disksep` is Dek's active list separator, which I use abundantly in macro writing. I call it the list element tag, because it is also needed before the first element.[11]

7 Flip-flop traversal

It occurs that loops are processed with the first or last traversal different from the others. In practice I needed—well, it was more elegant—in a play to handle only one player in each traversal but toggle the players: player, opponent, player, opponent, etc.

+	o	+
	o	+
o		+

The play at hand was tic-tac-toe, and the prototype implementation which demonstrates the toggling reads as follows.

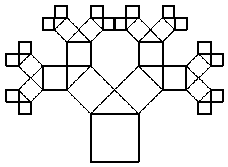
```
\def\play{\initialize
\loop\showboard
   \ifx\mark\markplayer\let\mark\markopponent
   \else\let\mark\markplayer
   \fi\iw{Supply index for \mark:}
      \read0to\index
   \ea\xdef\csname\index\endcsname{\mark}
\ifnum\index>0
\repeat}
\def\markplayer{+}\def\markopponent{o}
\endlinechar-1 %TB20.18
\play \bye
%with auxiliaries
%(\iw is shortcut for \immediate\write0)
\def\showboard{\iw{\1\2\3}...\iw{\7\8\9}}
\def\initialize{\def\1{-}...\def\9{-}}
```

In a general sense I need often in a repetitive situation to do something different at the beginning or at the end. Examples are my implementation of

- `\nitem`—numbered item and ilks
- to suppress the first headline of a chapter, and
- the `\\` in the linear sorting at the beginning.

8 Trees

Another intriguing repetitive or tail-recursive situation occurs when coding trees in TEX.[12] For fun—though the undertone is serious—I have added a tree from the Pythagorean family, borrowed from `pic.dat`.



The essence of the implementation in TEX of the repetition is shown below. For the full coding consult `pic.dat`.

```
\let\drawsq\ldrawsq %left drawing square
\def\pyth{\ifnum\level=1 \htyp\fi
    \drawsq\advance\level-1
    \multiply\kk18\divide\kk25
    {\turn7\x\leftx  \y\lefty
     \let\drawsq\ldrawsq\pyth}%
     \turn1\x\rightx \y\righty
     \let\drawsq\rdrawsq\pyth}
\def\htyp##1\pyth{\fi}
```

`\turn` turns over a multiple of 45°. (`\leftx`, `\lefty`) and (`\rightx`, `\righty`) contain the coordinates to start drawing a left square and a right square, respectively.

9 Conclusion

Many slightly different codings for a loop are around. This contributes to the reasons why understanding macro collections is so hard. It is so easy to come up with a variant, inhibiting intelligibility and trustworthiness.

I consider van der Goot's loop the simplest and most general, though Knuth's loop is usually sufficient for me.

10 Looking back

It is fun to flashback at for example Child's TEX Selftest, 1989, and to conclude that much of the perceived important TEXing nitty-gritties is not needed—not to mention all those essential issues which are lacking—when developing something like BLUe's format system.

This is precisely the reason why I pay so much attention to flashbacks, to summarize the paradigms, to bring to light what is needed in practical work, in such a way that the essence can be reused easily. Towards a discipline of TEXing. My case rest.

Have fun, and all the best

[11] I use them also with more than one argument, especially in selective loading of entries from a database. The above construction of a list is also used in my test example for sorting random words. Each character is obtained randomly and placed after the loop to form words of random length. Neat.

[12] Well also in METAFONT and PostScript.

Paradigms: Searching

Kees van der Laan

1 BLUe's Design VIII

Hi folks. I started using LaTeX for my hobby bridge, to typeset bid sequences and plays. Important in these kind of plays is data integrity, i.e. the system should remember that a card has been played. In TeXnical terms it comes down to update the computer memory. This is precisely what makes computer-assisted formatting different from previous formatting techniques. We can update memory.

This is a common phenomenon and the important building block is the handling of sets, that is trace an element and update the set, or extract from it, casu quo add to it.

The central macro in BLUe's format searching is called `\loc`, from locate.

Operations are to modify such as delete a card in bridge, or to extract, to copy the searched for element. The latter is applied abundantly in BLUe's format selective loading.

Below I'll discuss searching TeXniques, as used in The TeXbook, David Salomon's searching, and the various search activities which have been applied in BLUe's format.

2 Searching in TeX

Intrinsic in TeX is the searching for optimal linebreaks within a paragraph and for good pagebreaks of the main vertical list. Too complex material to be treated in this note.

3 Searching in METAFONT

A very nice feature is its capability to solve equations especially to determine intersection points of curves, specified declaratively. Searching via METAFONT will not be addressed in this paper.

4 Searching in The TeXbook

The examples of searching are about balancing columns.

On page 387 it is applied to adjusting the column widths when we have the English text in column one and the same text in another language in column two. With fixed column widths the column lengths are different.

On page 417 the manmac macro `\balancecolumns` is given to balance two columns on the last page of the index of The TeXbook. The latter TeXniques and macros have been used—and customized a little—for typesetting BLUe's format indexes.[1]

5 Salomon's procrusting

Salomon[2] looks for the optimal font size in fitting text to a box of prescribed size. The essence of the search process is given below stripped from other aspects. I modified his example into the following.

```
%Salomon's fitting text to a box
       %<TeX_Marker>
%Sept 95, cgl@rc.service.rug.nl
\begingroup
%Text
\toks0{Leentje leerde lotje lopen
       langs de lange lindenlaan.}
%Restrictions
\hsize=3cm \dimen1=20pt%\vsize=20pt
\def\showresult{\rlap{\f
    Fontsize=\the\dimen0\quad
    ht, dp=\the\ht0, \the \dp0}
    \copy0\smallbreak}
%Start: first estimate size of font
\dimen0=5pt
\loop\font\f=cmr10 at\dimen0
   \setbox0=\vbox{\tolerance10000
   \baselineskip\dimen0
   \f \the\toks0 }
\showresult
\ifdim\ht0<\dimen1 %next, linear search
   \advance\dimen0 by1pt
\repeat
%make the best fit
\advance\dimen0 by-1pt
\font\f=cmr10 at\dimen0
   \setbox0=\vbox to\dimen1{\tolerance10000
   \baselineskip\dimen0 plus 1pt minus1pt
   \f \the\toks0 }
\showresult
\endgroup

%Remark: extension to adjust for total size
%of box is left to the reader, and the
%context
\bye
```

Remark. For more advanced applications see Arsenau's macros 'Typesetting paragraphs of a specific shape,' MAPS 93.2.

6 Finding an element

The basic functionality is to locate an element in a set. The result of the process—success or failure—is stored in the 'Boolean' `\iffound`.

[1] My contribution in this area is that I process indexes on-the-ly, in one-pass.

[2] Advanced TeX, Springer, 1995, page 189.

[3] Assumed is that the set does not contain a period.

BLUe's format uses the following approach.[3]

```
\def\loc#1#2{%locate #1 in #2 a def
  \def\locate##1#1##2\end{\ifx\empty##2\empty
    \foundfalse\else\foundtrue\fi}%
  \ea\locate#2.#1\end}
```

To append the searched for element at the end of the arguments for `\locate` is related to the sentinel technique in programming.

Example (Printing vowels in bold)

Schwarz 1987 coined the problem to print vowels in bold face.[4]

The problem can be split into two parts. First, the general part of going character by character through a string, and second, decide whether the character at hand is a vowel or not.

For the first part use `\fifo`.

For the second part, combine the vowels into a string, `aeiou`, and the problem can be reduced to the question ⟨*char*⟩ ∈ `aeiou`?

With `\process` appropriately defined—locate the argument in the string of vowels—`\fifo Audacious\ofif` yields **Au**d**a**c**iou**s, with

```
\def\process#1{\uppercase{\loc#1}%
  {AEIOU}\iffound{\bf#1}\else#1\fi}
```

Example (Searching a set of accent strings)

In sorting in TEX I determine whether a control symbol belongs to the set of accents—`\def\accstr{\‘\’\"\^\c}`. In the macro `\nxtw` the relevant invoke reads as follows.

```
\ea\loc\head\accstr
```

The same approach has been applied in my BLUe-2-MAPS convertor assistant.

Example (Searching a set of Pascal reserved words)

In typesetting PASCAL fragments I determine whether a word belongs to the set of reserved ‘words.’ The set of reserved words reads

```
\reservedset{and array begin case const
 div do downto else end. end; file for
 function goto if in label mod nil not of
 or packed procedure program record repeat
 set then to type until var while with}
```

The relevant invoke in `\processw` reads

```
\loc{#1}{\the\reservedset}%
```

The Pascal environment scans the program fragment line-by-line and each line word-by-word. Each word is tested against the set of reserved words.

7 Deleting an element

The functionality is to delete an element from a set. It consists of two steps: finding and deleting. It can be coded as a variant of `\loc`, with as result not a Boolean but the modified set. A template might read as follows.

```
\def\delete#1#2{%Delete #1 from #2 a def
  \def\locate##1#1##2\end{\ifx\empty##2\empty
    \else\def#2{##1##2}\fi}%
  \ea\locate#2#1\end}
```

Addition of an element to a set can be done as follows.[5]

```
\def\add#1#2{%Add #1 to #2 a def
   \ea\def\ea#2\ea{#2#1}}
%or when #2 consists of unexpandable tokens
\def\add#1#2{%Add #1 to #2 a def
   \xdef#2{#2#1}}
```

Example (Updating a hand in bridge)

Bridge is an elimination play like most games. From the start elements are removed. The deletion of a card is a little special because we know that the card is there. The macro is called `\strip` and has been adapted. The hand is a definition instead of a token variable.

```
\def\strip#1#2{%Function:
% delete card value #1, is AKQJT9...2
% from #2, a def.
  \def\wis##1#1##2\siw{\gdef#2{##1##2}%
   \ifx#2\empty\gdef#2{--}\fi}%
  \ea\wis#2\siw}
```

Remarks. In the above example the searching (and deleting) is a side-effect of the printing. It is merely there to guarantee data integrity. If only attention is paid to the main issues, the searching would have been remained hidden, under a ‘pile of cards’. It is worth it to make some kind of library macro out of it—at least a template—ready for reuse.

Realize that the searched for element is supplied dynamically.

Example (Modifying `\dospecials`)

Inspired upon the Babel macros the following alternatives which also obey the locality character.

```
%Variant \bbl@add@specials
%No grouping, nor edef,
%assumed \dospecials, \sanitize are defined.
%
\let\sanitize\empty \let\dospecials\empty
%
\def\addspecials#1{\ea\def\ea
  \dospecials\ea{\dospecials\do#1}%
                 \ea\def\ea
  \sanitize\ea{\sanitize\makeother#1}
}
%
%Variant \bbl@remove@specials
%No grouping, nor edef,
%assumed \dodefault available
%
\let\dodefault\empty
%
```

[4] His solution mixes up the picking up of list elements and the process to be applied. Moreover, his nesting of `\if`-s in order to determine whether a character is a vowel or not, is not elegant.

[5] This is not so much about searching but added for your convenience.

```
\def\removespecials#1{%
   \def\do##1{\ifnum'#1='##1
              \else\nx\do\nx##1\fi}
   \edef\dospecials{\dospecials}
   \def\makeother##1{\ifnum'#1='##1
              \else\nx\makeother\nx##1\fi}
   \edef\sanitize{\sanitize}
\let\do\dodefault
\let\makeother\makeotherdefault
}
```

8 Dek's set macros

In The TEXbook the locating of an element—and delete it—is done as follows.

Dek provides `\remequivalent`,[6] The TEXbook 380, which uses a very general TEXnique. I'll untangle and recast the coding in the FIFO—First In First Out—notation.

Example (A set is just a list of defs)

Suppose the set reads `\def\set{\a\b\c}`. Then

```
%Assume #2 not empty
\def\remequivalent#1\from#2{%
   \def\process##1{\ifx#1##1\else\nx##1\fi}
   \xdef#2{\ea\fifo#2\ofif}}
\def\fifo#1{\ifx\ofif#1\ofif\fi
   \process{#1}\fifo}
\def\ofif#1\fifo{\fi}
```

Example (A set is a list of defs with list element tags)

Suppose the set reads `\def\set{\\\a\\\b\\\c}`. Then

```
%Assume #2 not empty
\def\remequivalent#1\from#2{%
   \def\\##1{\ifx#1##1\else\nx\\\nx##1\fi}%
   \edef#2{#2}}
```

Once we use list element tags we can also have more general elements, as Dek put it 'control sequences which are `\ifx`-equivalent to a given control sequence.'

Example (A set is a more general list of control sequences)

Suppose the set reads `\def\set{\\\a\\\b\\\c}`. Then

```
%Assume #2 not empty
\def\remequivalent#1\from#2{\let\given=#1%
\def\\##1{\ifx#1##1\else\nx\\\nx##1\fi}%
   \edef#2{\ea\fifola#2\alofif}}
\def\fifola\\#1#2{\ifx#1\given\else\nx\\\nx#1\fi
   \ifx#2\alofif\alofif\fi\fifola#2}
\def\alofif#1\alofif{\fi}
```

Finally, it can be extended by the test for the emptiness of `\set`.

9 Database use

Databases are all about easy input, conveniently storing and retrieving, and appropriately reporting.

While developing BLUe's format system and using the database idea for storing and retrieving other search TEXniques have been coded. Needed are facilities for browsing a database next to macros for selective loading of addresses, a format, pictures, references, or tools.

The browse macros are based on `\loc`, and can be associated with keyword pattern recognition.

The selective loading macros are based on the proper definition of the list element tag.

Below the searching aspects have been put together. 'BLUe's Databases' treats the use and coding of BLUe's format databases in depth.

9.1 Pattern recognition

A nice application is mail-merge, merging addresses with a letter.

Example (Match addresses for the pattern RUSSIA)

```
\input blue.tex \letter \searchfile{address}
\search{RUSSIA}
\bye
```

To send a letter to all those persons or to make all address labels insert `\makesearchletters`, respectively `\makesearchlabels` before `\bye`.

Example (Coding the search macro)

At the heart lies the earlier mentioned `\loc` macro. Because we need to do more with found entries than just knowing that they are there, their names, preceded by the list element tag for further action, are collected in the toks variable `\namelst`. For convenience the names are also written to the log file, in order that we can follow what is going on.

```
\def\search#1{\def\loc##1##2{%
   \def\locate####1##1####2\end
   {\ifx\empty####2\empty\foundfalse
   \else\foundtrue\fi}\ea\locate##2.##1\end}
   \def\lst##1##2{\loc{#1}{##2}\iffound
   \immediate\write16{\nx##1}%log file
   \namelst\ea{\the\namelst\lst##1}
   \def##1{##2}%define found element
   \fi}\input\the\searchfile.dat\relax}
```

9.2 Selective loading

This is all about organizing collections and reusing parts of it. More restricted it is about using memory space economically, just to load what is needed, trading time—selection process and loading—for space.

Selective loading is also a search activity. The file is scanned and when an entry is found it will be loaded. The details of the selective loading process depend on the entries of the database, how they are stored. I distinquish class I and class II (TEX) databases. The first class consists of formats and tools—which could have been loaded non-selectively—and the second class consists of addresses, pictures, and references, for which selective loading is essential.

[6] As part of a suite of set macros.

9.2.1 Formats and tools

The idea is that for example `\letter` only loads the letter macros from `fmt.dat`. In the examples below use is made of `\tool`, with the functionality that when the tag after is known the control sequences which follow the tag up to `\endinput` will be loaded, otherwise all in between the tag and `\endinput` will be gobbled.

```
\long\def\tool#1{\ifx#1\undefined
          \bgroup\unouterdefs
            \ea\gobbletool\fi}
\long\def\gobbletool#1\endinput{\egroup}
```

Example (Coding the loading of the letter macros)

```
%From blue.tex
\def\letter{\ifx\undefined\letterfmt
   \let\letterfmt=x\fi
   \loadformat
   \let\letterfmt\undefined}
\def\loadformat{\input fmt.dat \relax}
%from fmt.dat
\tool\letterfmt
<lettermacros>
\endinput
```

Similarly, a tool can be loaded as follows.

Example (Coding the loading of the smiley macros)

```
%From blue.tex
\newbox\smileybox
\newcount\smileysloadcount
\def\smiley{\loadsmileys\raise.5ex
   \hbox{\unitlength.01pt
   \copy\smileybox
   \eyes\mouth
   \kern1000\unitlength} }
\def\winksmiley{<similar>}
\def\sadsmiley{<similar>}
\def\loadsmileys{\ifx\undefined\smileystool
   \let\smileystool=x\fi
   \ifnum\smileysloadcount=0 \ea\loadtool
   \else\message{--- smileys already
                  loaded---}\fi
   \advance\smileysloadcount1
   \let\smileystool\undefined}
%from tools.dat
\tool\smileystool
<smileymacros>
\endinput
```

Remark. Whenever suited a load counter is maintained such that double loading is inhibited.

9.2.2 Addresses, pictures and references

The entries of these database obey the syntax

`\lst\<name>{<entryproper>}`

The selective loading comes down to a proper definition of `\lst`. Moreover, the names of the entries to be selected must be defined with whatever you wish as replacement text.[7]

```
\def\loadselectivefrom#1{%#1 lit etc.
 \def\lst##1{\ifx##1\undefined\ea\gobble
         \else \ea\gdef\ea##1\fi}
 \input #1.dat \relax%e.g. lit.dat
 }
\def\gobble#1{}
```

Because of the scanning `\outer` defs are not allowed, nor are `\par`-s. The selective loading macro is embedded in the user macros `\references` and its ilks. In detail the meaning of 'loading' is adapted to the application. For references this means that the specified entries are set in a box and their names redefined by numbers. The names can be used for cross-referencing purposes while the box can be pasted up at your place of choice. However, the underlying searching methodology is the same for addresses, references and pictures.

Example (Coding the handling of references)

```
\def\references#1{\beginreferences#1%
   \endreferences}
\def\bluereferences#1\par
   {\beginreferences#1\endreferences}
\def\loadreferences{\loadselectivefrom
   {\the\referencesfile}}
\def\beginreferences#1\endreferences{%
  \bgroup\def\process##1{\ifx\undefined##1
     \global\let##1\referenceserror\else
     \message{***\tt\string##1
             already loaded.***}
  \fi\namelst\ea{\the\namelst\lst##1}}%
  \fifo#1\ofif
  \if]#1]\else\ea\loadreferences\fi
  %formatting
  \ifstore\global\setbox\referencesbox=
    \vbox\bgroup\fi\prenum{}\postnum{}
  \lsams%Default ls
  \the\thisreferences
  \def\lst##1{\ls{##1}
    \xdef##1{\the\itemno}}
  \the\namelst\endreferences}
```

Remark. BLUe's format style of coding is centred along two-part macros with a one-part macro on top, enriched by a convenient alias such as `\bluereferences`, for the user interface.

9.2.3 Max Díaz' TEXnique

The TEXbook 382–384 mentions the fast loading TEXnique of Max Díaz, which requires that every line is preceded by a special character. The process comes down to the following.

```
%The TeXbook, Appendix D 4.Selective loading.
%The Max Diaz fast selective loading process.
%(A little simplified, and combined with
% my list element tag, \lst.)
\def\lst#1{\catcode`\ =%tilde
   \ifx#1\undefined14    %comment
             \else9 \fi}%ignore char
%We want to load the cgl part.
\def\cgl{<anything>}

\lst\name
~\ a
~\ b
~\ c
\lst\cgl
~\ aa
~\ bb
~\ cc
\lst\erik
```

[7] This approach is the opposite of preventing reloading. We tacitly want to redefine the fancy entries by the meaningful ones. My fancy replacement text is an error message.

```
~\ aaa
~\ bbb
~\ ccc
\catcode`\~=13
%
<Commonpart>
\bye
%Explanation: the list element tag toggles
%the catcode for ~ such that either the
%first character is ignored (and the rest
%of the line loaded) or the line is a
%comment line.
```

In the above one can replace `\lst\name...` by `\input <filetobeloadedfrom>`.

9.2.4 Variant document parts

The idea is that a script is marked up also with markup tags which have a selection/omission function. For example an abridged version is interspersed within the script. The idea is that the owner can either ask for an abridged or a full version. Another example is documentation with details for various computer operating systems. Given a customer with a specific operating system only the relevant parts will be printed.[8]

With the new hype HTML this functionality may enjoy a second youth.

10 Tree search

When I implemented trees in TEX especially the Pythagorean trees—to illustrate turtle graphics—I played a little longer with it and could use TEX in 'dialogue mode' to search for a name by answering questions.

10.1 Interactive path through a binary tree

The following is inspired upon Greene's 'Playing in TEX's mind.'[9]

```
%Guess what? August 1995, cgl@rc.service.rug.nl
%Idea biased by A.M.Greene's
%Playing in TeX's mind, TUG 89.
%Interactive binary tree traversal.
%Interactive TeX ing,
%TeX as an engine to play with.
\input blue.tex
\def\bintree{\message{\csname\node\endcsname}
   \ea\ifx\csname\node0\endcsname\relax\eertnib\fi
   \read0to\yorn
   \edef\node{\node\if n\yorn1\else0\fi}
   \bintree}
\def\eertnib#1\bintree{\fi\def\node{1}
   \immediate\write0{Hope this is the one
               you are looking for :-) }
   \immediate\write0{}
   \message{Another play?}
   \read0to\yorn
   \if y\yorn\ea\bintree\fi
   \immediate\write0{Thank you, bye}}
%Rules of the game
\immediate\write0{Guess game.
 The system asks questions to be answered}
\immediate\write0{by *** y or n ***}
\immediate\write0{The following play
             guesses an NTG member}
\immediate\write0{}
%
%Data (a tree structure)
\begingroup\obeylines
\def\lst#1 #2
   {\ea\def\csname#1\endcsname{#2}}
\lst 1 NTG member?
\lst 10 Plain TeX ie?
\lst 100 Honoured?
\lst 1000 Kees
\lst 1001 HH
\lst 101 On board?
\lst 1010 Chair?
\lst 10100 Erik
\lst 10101 Secretary?
\lst 101010 Gerard
\lst 101011 Treasurer?
\lst 1010110 Wietse
\lst 1010111 Dark?
\lst 10101110 Johannes
\lst 10101111 Frans
\lst 1011 Anonymous
\lst 11 Just a friend
%
%Start the play
\endlinechar-1 %TB20.18
\def\node{1}\bintree
\endgroup
%
%Typesetting the data
\onecol
Pretext
\thisbt{\xoffset{-400}}
{\obeylines
\beginbt1 NTG member?
10 Plain TeX ie?
100 Honoured?
1000 cgl
1001 HH
101 On board?
1010 Chair?
10100 Erik
10101 Secretary?
101010 Gerard
101011 Treasurer?
1010110 Wietse
1010111 Dark?
10101110 JLB
10101111 FG
1011 Anonymous
11 Just a friend
8 \endbt
}Posttext
\bye
```

A typical log file looks as follows.

```
Guess game. The system asks questions to
be answered by *** y or n ***
The following play guesses an NTG member

NTG member?
\yorn=y
 Plain TeX ie?
\yorn=y
 Honoured?
\yorn=y
 Kees
Hope this is the one you are looking for :-)

Another play?
\yorn=y
 NTG member?
\yorn=n
 Just a friend
Hope this is the one you are looking for :-)
```

[8] In real-life this is hardly done. When I buy a TV the operation booklet contains the information in several languages.

[9] Courtesy Bernd Raichle for node representation.

```
Another play?
\yorn=y
 NTG member?
\yorn=y
 Plain TeX ie?
\yorn=n
 On board?
\yorn=y
 Chair?
\yorn=y
 Erik
Hope this is the one you are looking for :-)

Another play?
\yorn=y
 NTG member?
\yorn=y
 Plain TeX ie?
\yorn=n
 On board?
\yorn=y
 Chair?
\yorn=n
 Secretary?
\yorn=y
 Gerard
Hope this is the one you are looking for :-)

Another play?
\yorn=y
 NTG member?
\yorn=y
 Plain TeX ie?
\yorn=n
 On board?
\yorn=n
 Anonymous
Hope this is the one you are looking for :-)

Another play?
\yorn=n
```

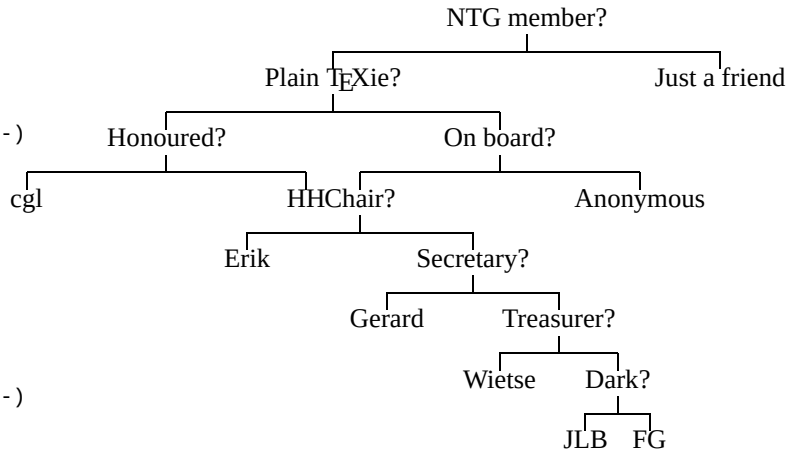
Remarks. The first version used a counter to represent the nodes with the restriction of 2^{16}.

Robustness with respect to mistypes of the user after the prompt—the user does not supply ‘y’ or ‘n’—has been treated in Paradigms: Loops.

10.2 The tree

The typesetting of the binary tree visualizes the data for your convenience. Within BLUe’s format this goes as follows

```
\thisbt{\xoffset{-400}}
\beginbt 1 NTG member?
10 Plain \TeX ie?
...
11 Just a friend
8 \endbt
```



Have fun, and all the best

Paradigms: Sorting

Kees van der Laan

1 BLUe's Design IX

Hi folks. A strong and *unique* point of BLUe's format system is its indexing on the fly. Be it for a total document or just for a chapter. One of the requisites for indexing on the fly is the possibility to sort within TEX.

Sorting has always been an important topic in computer science. In TEX I needed sorting on several occasions especially for sorting numbers such as citation lists, words such as addresses, and index entries.

This note is devoted to paradigms encountered while implementing and applying sorting in TEX.

Sorting can be characterized by
- the set to be sorted (numbers, word. etc.)
- the addressing of elements of the set
- the ordering for the set
- the comparison operation, and
- the exchange operation.

To do some sorting of your own please load from `blue.tex` the index macros via `\loadindexmacros`. Below parts have been extracted from that collection of macros to make this note as intelligible as possible. `\ea` is my shortcut for `\expandafter`.

2 Linear sorting

A simple sorting method is repeatedly searching for the smallest element. In the example below the set is defined as a def with list element tag `\\`.

```
\def\lst{\\\ia\\\ib\\\ic}
\def\ia{314} \def\ib{1} \def\ic{27}
%
\def\dblbsl#1{\ifnum#1<\min\let\min=#1\fi}
%
\loop\ifx\empty\lst\expandafter\break\fi
   \def\\{\let\\=\dblbsl\let\min=} %space
   \lst%find minimum
   \min%typeset minimum
   {\def\\#1{\ifx#1\min \else\nx\\%
    \nx#1\fi}\xdef\lst{\lst}}%
\pool%Inspired upon van der Goot's
     %Midnight macros.
\def\loop#1\pool{#1\loop#1\pool}
\def\break#1\pool{}
```

The coding implements the looping of the basic steps
- find minimum (via `\lst`, and suitable definition of DeK's list element tag `\\`)
- typeset minimum (via `\min`)
- delete minimum from the list (via another appropriate definition of the list element tag, and the use of `\xdef`.

Remark. The kludge for using `\ifx` instead of `\ifnum` in the deletion part is necessary because TEX inserts a `\relax`.

3 Sorting in an array

If we adopt array addressing in TEX for the elements to be sorted then we can implement bubble sort in TEX too.[1]

3.1 Array addressing

When we think of associating values to (index) numbers—`1` → `\value{1}` —then we are talking about an array. A mapping of the natural numbers on ... for example the natural numbers. The `\value` control sequence can be implemented as follows.

```
\def\value#1{\csname#1\endcsname}
```

The writing to the array elements can be done via

```
\def\1{<value1>} \def\2{<value2>}...
```

In general this must be done via

```
\ea\def\csname<number>\endcsname{<valuenumber>}
```

3.1.1 To get the hang of it

The reader must be aware of the differences between
- the index number, $\langle k \rangle$
- the counter variable `\k`, with the value $\langle k \rangle$ as index number
- the control sequences `\<k>`, $k = 1, 2, \ldots, n$, with as replacement texts the items to be sorted.

When we have `\def\3{4} \def\4{5} \def\5{6}` then
`\3` yields **4**,
`\csname\3\endcsname` yields **5**, and
`\csname\csname\3\endcsname\endcsname` yields **6**.

Similarly, when we have
`\k3 \def\3{name} \def\name{action}` then
`\the\k` yields **3**,
`\csname\the\k\endcsname` yields **name**, and
`\csname\csname\the\k\endcsname\endcsname` yields **action**.[2] To exercise shortcut notation the last can be denoted by `\value{\value{\the\k}}`.

[1] The above example of linear sorting can be seen as sorting in a so-called associative array.

[2] Confusing, but powerful.

Another `\csname...` will execute `\action`, which can be whatever you have provided as replacement text.[3]

4 Bubble sort

This process looks repeatedly for the biggest element which is swapped to the end. This is done for the complete array, the array of size $n-1$ et cetera. The pseudo code reads as follows.

for $n :=$ *upb* **downto** 2 **do**

begin for $k :=$ $n-1$ **downto** 1 **do**

if $a[n] < a[k]$ **then**

exchange(a[*n*], *a*[*k*]*)*;

end;

The TEX macro reads as follows.

```
\def\bubblesort{%Data in defs \1, \2,...\<n>.
%Result: \1<=\2<=...<=\<n>.
{\loop\ifnum1<\n{\k\n
  \loop\ifnum1<\k \advance\k-1
     \cmp{\deref\k}{\deref\n}%
   \ifnum\status=1 \xch\k\n\fi
  \repeat}\advance\n-1
 \repeat}}%end \bubblesort
%with auxiliaries
\def\deref#1{\csname\the#1\endcsname}
\let\cmp\cmpn %from blue.tex or provide
%\def\cmp#1#2{%Comparison. Yields
% \status=0, 1, 2 for =, >, <
%...}
%\def\xch#1#2{%exchange
%#1, #2 counter variables
%...}
```

5 Heap sort

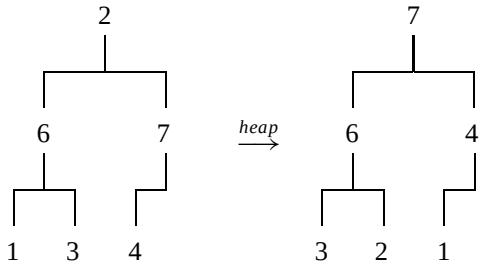
We can organize the array as a heap. A heap is an ordered tree. Loosely speaking for each node the siblings are smaller or equal than the node.

The process consists of two main steps
- creation of a heap
- sorting the heap

with a sift operation to be used in both.

In comparison with my earlier release of the code in MAPS 92.2, I adapted the notation with respect to sorting in *non-decreasing* order.[4]

What is a heap? A sequence $a_1, a_2, \ldots, a_n$, is a heap if $a_k \geq a_{2k} \wedge a_k \geq a_{2k+1}, k = 1, 2, \ldots, n \div 2$, and because a_{n+1} is undefined, the notation is simplified by defining $a_k > a_{n+1}, k = 1, 2, \ldots, n$.
A tree and one of its heap representations of 2, 6, 7, 1, 3, 4 read



In PASCAL-like notation the algoritm, for sorting the array a[1:n], reads

{ heap creation }

$l :=$ n **div** 2 $+$ 1;

while $l \neq 1$ **do**

begin $l :=$ $l-1$; *sift(a, l, n)* **end**;

{ sorting }

$r :=$ n;

while $r \neq 1$ **do**

begin *swap(a*[1], *a*[*r*]*)*;

$r :=$ $r-1$; *sift(a*, 1, *r)*

end;

{ sift arg1 through arg2 }

$j :=$ *arg*1;

while $2j \geq$ *arg*2 **and**

(a[*j*] $<$ *a*[2*j*] **or** *a*[*j*] $<$ *a*[2*j* + 1]*)*

do begin *mi* $:=$ $2j$ $+$ **if** *a*[2*j*] $>$ *a*[2*j* + 1]

then 0 **else** 1;

exchange(a[*j*], *a*[*mi*]*)*; $j :=$ *mi*

end;

5.1 Purpose

Sorting values given in an array.

5.2 Input

The values are stored in the control sequences `\1`, ..., `\<n>`. The counter `\n` must contain the value ⟨*n*⟩. The parameter for comparison, `\cmp`, must be `\let`-equal to
- `\cmpn`, for numerical comparison,
- `\cmpw`, for word comparison,

[3] My other uses of the `\csname` construction are: to let TEX accept an outer defined macro name in a replacement text, to check whether a name has already been defined, and to mimic a switch selector.

[4] It is true that the reverse of the comparison operation would do, but it seemed more consistent to me to adapt the notation of the heap concept with the smallest elements at the bottom.

- `\cmpaw`, for word comparison obeying the ASCII ordering, or
- a comparison macro of your own.

5.3 Output

The sorted array `\1`, `\2`, ... `\<n>`, with `\value1` ≤ `\value2` ≤ ... ≤ `\value`⟨*n*⟩.

5.4 Source

```
%Non-descending sorting
\def\heapsort{%data in \1 to \n
\r\n\heap\ic1
{\loop\ifnum1<\r\xch\ic\r
   \advance\r-1 \sift\ic\r
\repeat}}
%
\def\heap{%Transform \1..\n into heap
 \lc\n\divide\lc2{}\advance\lc1
 {\loop\ifnum1<\lc\advance\lc-1
  \sift\lc\n\repeat}}
%
\def\sift#1#2{%#1, #2 counter variables
 \jj#1\uone#2\advance\uone1 \goontrue
 {\loop\jc\jj \advance\jj\jj
  \ifnum\jj<\uone
   \jjone\jj \advance\jjone1
   \ifnum\jj<#2 \cmpval\jj\jjone
         \ifnum2=\status\jj\jjone\fi\fi
   \cmpval\jc\jj\ifnum2>\status\goonfalse\fi
 \else\goonfalse\fi
\ifgoon\xch\jc\jj\repeat}}
%
\def\cmpval#1#2{%#1, #2 counter variables
%Result: \status= 0, 1, 2 if
%values pointed by
%              #1 =, >, < #2
 \ea\let\ea\aone\csname\the#1\endcsname
 \ea\let\ea\atwo\csname\the#2\endcsname
 \cmp\aone\atwo}
%
\def\cmpn#1#2{%#1, #2 must expand into
             %numbers
%Result: \status= 0, 1, 2 if
%        \val{#1} =, >, < \val{#2}.
 \ifnum#1=#2\global\status0 \else
  \ifnum#1>#2\global\status1 \else
            \global\status2 \fi\fi}
%
\def\xch#1#2{%#1, #2 counter variables
 \edef\aux{\csname\the#1\endcsname}\ea
 \xdef\csname\the#1\endcsname{\csname
   \the#2\endcsname}\ea
 \xdef\csname\the#2\endcsname{\aux}}.
%with auxiliaries
\newcount\n\newcount\lc\newcount\r
\newcount\ic\newcount\uone
\newcount\jc\newcount\jj\newcount\jjone
\newif\ifgoon
```

Explanation.

`\heapsort` The values given in `\1`,... `\<n>`, are sorted in non-descending order.

`\heap` The values given in `\1`,... `\<n>`, are rearranged into a heap.

`\sift` The first element denoted by the first (counter) argument has disturbed the heap. Sift rearranges the part of the array denoted by its two arguments, such that the heap property holds again.

`\cmpval` The values denoted by the counter values, supplied as arguments, are compared.

Example (Numbers, words)

`\cmpn`, and `\cmpw` stand for compare numbers and words. `\prtn`, and `\prtw` stand for print numbers and words, and work the way you expect. `\accdef` takes care that accents are properly defined.

```
\def\1{314}\def\2{1}\def\3{27}\n3
\let\cmp\cmpn\heapsort
\beginquote\prtn,\endquote
%
\def\1{ab}\def\2{c}\def\3{aa}\n3
\let\cmp\cmpaw\heapsort
\beginquote\prtw,\endquote
and
\def\1{j\ij}\def\2{ge\"urm}\def\3{gar\c con}
\def\4{\'el\'eve}\n4
\let\cmp\cmpw {\accdef\heapsort}
\beginquote\prtw\endquote
```

yields within the context of `blue.tex`

1, 27, 314,

aa ab c,

and élève, garçon, geürm, jij.

6 Quick sort

The quick sort algorithm has been discussed in many places, The following code is borrowed from Bentley.[5]

procedure *QSort(low,up)*;

if *low* < *up* **then**

begin

{ choose suitable median }

Swap(X[*low*], *X*[*RandInt(low,up)*]);

T := *X*[*low*]; *M* := *low*;

{ Invariant loop

X[low + 1..M] < T and X[M + 1..I − 1] ≥ T }

for *I* := *low* + 1 **to** *up* **do**

if *X*[*I*] < *T* **then**

begin *M* := *M* + 1;

Swap(X[*M*], *X*[*I*]);

end;

{ exchange median }

Swap(X[*low*], *X*[*M*]);

{ X[low..M − 1] < X[M] ≤ X[M + 1..up] }

QSort(low, M − 1); *QSort(M* + 1, *up)*;

end;

[5] Programming Pearls, Addison-Wesley. It contains also diagrams which keep track of the invariants.

6.1 Purpose

Sorting of the values given in the array \<low>, ..., \<up>.

6.2 Input

The values are stored in \<low>, ..., \<up>, with $1 \leq low \leq up \leq n$. The parameter for comparison, \cmp, must be \let-equal to

- \cmpn, for number comparison,
- \cmpw, for word comparison,
- \cmpaw,for word comparison obeying the ASCII ordering, or
- a comparison macro of your own.

6.3 Output

The sorted array \<low>, ..., \<up>, with \val⟨*low*⟩ ≤ ... ≤ \val⟨*up*⟩.

6.4 Source

```
\def\quicksort{%Values given in
%\low,...,\up are sorted, non-descending.
%Parameters: \cmp, comparison.
 \ifnum\low<\up\else\brk\fi
%\refval, a reference value selected
%at random.
 \m\up\advance\m-\low%Size-1 of array part
 \ifnum10<\m\rnd\multiply\m\rndval
   \divide\m99 \advance\m\low \xch\low\m
 \fi
 \ea\let\ea\refval\csname\the\low\endcsname
 \m\low\k\low\let\refvalcop\refval
 {\loop\ifnum\k<\up\advance\k1
   \ea\let\ea\oneqs\csname\the\k\endcsname
   \cmp\refval\oneqs\ifnum1=\status
      \global\advance\m1 \xch\m\k\fi
   \let\refval\refvalcop
  \repeat}\xch\low\m
 {\up\m\advance\up-1 \quicksort}%
 \low\m\advance\low1 \quicksort}
%
\def\brk#1\quicksort{\fi}
```

Explanation. At each level the array is partitioned into two parts. After partitioning the left part contains values less than the reference value and the right part contains values greater than or equal to the reference value. Each part is again partitioned via a recursive call of the macro. The array is sorted when all parts are partitioned.

In the TEX coding the reference value as estimate for the mean value is determined via a random selection of one of the elements.[6] Reid's \rnd has been used. The random number is mapped into the range [*low* : *up*], via the linear transformation \low + (\up − \low) ∗ \rndval/99.[7]

The termination of the recursion is coded in a TEX peculiar way. First, I coded the infinite loop. Then I inserted the condition for termination with the \fi on the same line, and not enclosing the main part of the macro. On termination the invocation \brk gobbles up all the tokens at that level to the end, to its separator \quicksort, and inserts its replacement text, a new \fi, to compensate for the gobbled \fi.

6.5 Auxiliaries

Sorting is parameterized by comparison and exchanging. Also needed is a random number generator. The latter is not supplied here.

```
\def\cmpn#1#2{%#1, #2 must expand into
              %numbers
%Result: \status= 0, 1, 2 if
%        \val{#1} =, >, < \val{#2}.
 \ifnum#1=#2\global\status0 \else
  \ifnum#1>#2\global\status1 \else
             \global\status2 \fi\fi}
%
\def\xch#1#2{%#1, #2 counter variables
 \edef\aux{\csname\the#1\endcsname}\ea
 \xdef\csname\the#1\endcsname{\csname
   \the#2\endcsname}\ea
 \xdef\csname\the#2\endcsname{\aux}}
```

6.6 Ordering

The ordering is parameterized in the ordering table.

Example (Numbers, words)

\cmpn, and \cmpw stand for compare numbers and words. \prtn, and \prtw stand for print numbers and words, and work the way you expect. \accdef takes care that accents are properly defined.

```
\def\1{314}\def\2{1}\def\3{27}\n3
\low1\up\n\let\cmp\cmpn
\quicksort
\beginquote\prtn,\endquote
%
\def\1{ab}\def\2{c}\def\3{aa}
\def\4{\ij}\def\5{ik}\def\6{z}\def\7{a}\n7
\low1\up\n\let\cmp\cmpw
\quicksort
\beginquote\prtw,\endquote
and
\def\1{j\ij}\def\2{ge\"urm}\def\3{gar\c con}
\def\4{\'el\'eve}\n4
\low1\up\n\let\cmp\cmpw
{\accdef\quicksort}
\beginquote\prtw.\endquote
```

yields similar results as with heap sorting.

7 Use

I needed sorting within TEX for indexing and for sorting address labels.

7.1 Sorting address labels

Suppose we wish to sort addresses on the secondary key membership number. In order to do so the index must point to the name of the database entry and the name must point to its membership number, that is

$$1\,2\,\ldots \rightarrow \texttt{<name>}_x\ \texttt{<name>}_y\ \ldots \rightarrow \texttt{<no>}_x\ \texttt{<no>}_y\ \ldots$$

This can be coded as follows.

[6] If the array is big enough. I chose rather arbitrarily 10 as threshold.

[7] Note that the number is guaranteed within the range.

```
\loadindexmacros
%
\def\lst#1#2{\advance\k1
   \ea\def\csname\the\k\endcsname{#1}%
   \ea\def\ea#1\gobbletono#2}
\def\gobbletono#1\no{}
\k0
\input toy.dat %The test database
\n\k           %number of items
Membershipno unsorted: \1, \2, ...
%
\let\cmp\cmpn\sort

Sorted on membershipno: \1, \2, ...
```

The amazing thing is that we don't have to do much extra because the name will expand to the number, which will be used in the comparison. I used that `\no` was the last element of the database entry, but that is not essential. Each database entry consist of a triple `\lst`, `\<name>`, and entry proper within braces.

7.1.1 Typesetting

Now we have to redirect the pointer from the name away from the number to the complete entry, that is

$$1\,2\,\ldots \rightarrow \texttt{\<name>}_1\ \texttt{\<name>}_2\ \ldots \rightarrow \texttt{entry}_1\ \texttt{entry}_2\ \ldots$$

This is done as follows.

```
\def\lst#1#2{\def#1{#2}}
\input toy.dat
\1 \2 \3 \4 \5 \6
```

7.2 Sorting index entries

One of the processes in preparing an index is sorting the Index Reminders, IRs. This is again a sorting process on secondary keys, even tertiary keys.

Given the sorting macros we just have to code the special comparison macro in compliance with `\cmpw`: compare two 'values' specified by `\def`s. Let us call this macro `\cmpir`.[8] Each value is composed of

- a word (action: word comparison)
- a digit (action: number comparison), and
- a page number (action: (page) number comparison).

The macros read as follows.

```
\def\cmpir#1#2{%#1, #2 defs
%Result: \status= 0, 1, 2 if
%        \val{#1} =, >, < \val{#2}
 \ea\ea\ea\decom\ea#1\ea;#2.}
%
\def\decom#1 !#2 #3;#4 !#5 #6.{%
 \def\one{#1}\def\four{#4}\cmpaw\one\four
 \ifnum0=\status%Compare second key
   \ifnum#2<#5\global\status2 \else
     \ifnum#2>#5\global\status1 \else
       %Compare third key
       \ifnum#3<#6\global\status2
       \else\ifnum#3>#6\global\status1 \fi
       \fi
     \fi
   \fi
 \fi}
```

Explanation. I needed a two-level approach. The values are decomposed into their components by providing them as arguments to `\decom`.[9] The macro picks up the components

- the primary keys, the ⟨*word*⟩
- the secondary keys, the ⟨*digit*⟩, and
- the tertiary keys, the ⟨*page number*⟩.

It compares the primary keys, and if necessary successively the secondary and the tertiary keys. The word comparison is done via the already available macro `\cmpaw`.

To let this work with `\sort`, we have to `\let`-equal the `\cmp` parameter to `\cmpir`.

8 Sorting in the mouth

Alan Jeffrey and Bernd Raichle have provided macros for this. The following variant of the linear sorting given at the beginning of this note is inspired upon Bernd's 'Quick Sort in the Mouth,' EuroTEX 94. The idea is that a sequence is split in its smallest element and the rest by an invoke of `\fifo`. The rest is treated recursively as a similar sequence. Another example of (multiple) nested FIFO.

```
\def\fifo#1%accumulated rest
         #2%smallest
         #3%next
{\ifx\ofif#3 #2\ofif{#1}\fi
 \ifnum#3<#2
   \p{\fifo{#1{#2}}{#3}}\else
   \q{\fifo{#1{#3}}{#2}}\fi}
%repeat or terminate
\def\ofif#1\fi#2\fi{\fi
   \if*#1*\endsort\fi
   \fifo{}#1\ofif}
%auxiliaries
\def\p#1\else#2\fi{\fi#1}
\def\q#1\fi{\fi#1}
%terminator
\def\endsort#1\ofif{\fi}
%test
\fifo{}3{123}8{1943}\ofif
```

To assure yourself that it is all done in the mouth `\write` the test.[10]

However, in sorting within TEX I prefer a uniform approach not in the least parameterized over the ordering table.

Have fun, and all the best

[8] Mnemonics: compare index reminders

[9] Mnemonics: decompose. In each comparison the defs are 'dereferenced,' that is their replacement texts are passed over. This is a standard TEXnique: a triad of `\ea`s, and the hop-overs to the second argument.

[10] I don't know how to ensure correctness. It is tricky to get the braces right. I used `\tracingmacros=1`.

Paradigms: Just a little bit of PostScript

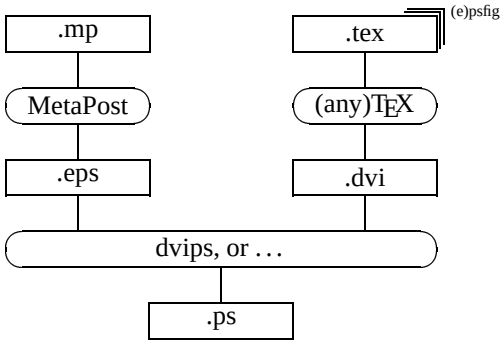
Kees van der Laan

Abstract

It is all about creating EPS—with graphics—to be merged with (La)TEX scripts. The emphasis is on creating raw PostScript for simple symmetrical pictures. Asides, like incorporating accurate graphs of math functions, typesetting text along curved paths, or tables set sideways, next to reverse video, clipping and tiling have been addressed. A poor man's mftoeps approach is touched upon: (declarative) METAFONT into (imperative) PostScript.

1 BLUe's Design X

Hi folks. The user's guide which comes with BLUe's format system—Publishing with TEX, PWT for short—is processed *completely* by TEX, *no* other tools such as POSTSCRIPT are needed.[1] However, of late I exercised METAFONT—well, eventually MetaPost with the help of Jos Winnink—for graphics to be included in TEX documents, and finally embarked PostScript straightaway to create EPS pictures, with the help of Joseph Romanovsky.



POSTSCRIPT is involuntary needed to (electronically) paste up the graphics, and as resulting file format.[2]

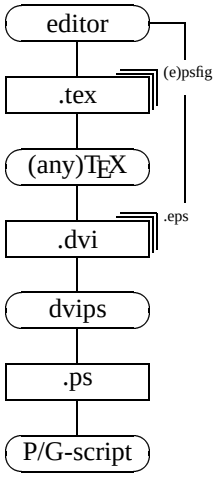
If we come to think of graphics as

just doing the 'right' strokes or fills

then POSTSCRIPT provides the means for this: lines, splines and circular arcs, to be drawn or filled.[3] I use the sidestep

METAFONT → MetaPost → EPS

for general pictures but also for obtaining the right (control) points explicitly from a decalarative specification in METAFONT, as shown by Escher's knot at the end.[4]



With respect to graphics POSTSCRIPT can be seen as a *little* language in the UNIX tradition.[5] A little bit of POSTSCRIPT adheres the 80%–20% adage: 80% of the effects (or more) with 20% of the energy (or less).

One can with a little knowledge of POSTSCRIPT code graphics immediately and *completely* in POSTSCRIPT. The more so because of the ubiquitous public domain GhostScript previewers to verify the result, next to of course the POSTSCRIPT laser printers.

Furthermore, text is just a special case of graphics, and merging just a little bit of text—malenki Russians would say—with the graphics goes equally simple at first glance.

And to end the lovesong the inclusion of accurate graphs of mathematical functions goes well via coding these in

[1] Nobody knows what the future has in store, but for the moment I consider it a good thing that the PWT guide can be processed just by TEX, well . . . with BLUe's format.

[2] For exchange the `.tex` and (hand-coded) `.eps` files are much better suited because of their conciseness. This can't be beaten, not even by Adobe's PDF—Portable Document Format.

[3] PostScript II also provides for colors and processing in a network.

[4] Another way for arriving at the EPS code is to use Jackowski's `mftoeps`package or to use MetaPost.

[5] See Bentley's Little languages in 'More programming pearls—Confessions from a coder.' Addison-Wesley.

POSTSCRIPT and including these as figures. (Of course Hobby's graph extension could be used as well, or other advanced graphics packages.) This is illustrated by a graph of the sine function to convey the idea.

PStricks is about *interfacing*. Not assuming knowledge of POSTSCRIPT. This note discusses mainly *merging*. Is about extending your TEXpertise with just a little—tsjut-tsjut Russians would say—knowledge of POSTSCRIPT rewarded by high returns.

Below I'll summarize what is needed from POSTSCRIPT, and illustrate the use of it with a few examples, introducing en route the operators we need given the example.

2 PostScript

2.1 Processing

POSTSCRIPT comes with a user's guide (cookbook) and reference manual, the so-called blue and red books in the Adobe POSTSCRIPT series. For processing POSTSCRIPT an interpreter is needed, such as a POSTSCRIPT laser printer or a GhostScript previewer. For inclusion in (La)TEX I use the psfig macros.[6] Goossens in his PostScript and (La)TEX, MAPS 92.1, nicely details about inclusion of PostScript.[7]

As with PDF I consider the post-processing capability *independently* from the tool which created the POSTSCRIPT source, very powerful and flexible.

2.2 Why writing PostScript?

History has it that POSTSCRIPT programs are not written by humans but generated by high-level tools, such as MetaPost. This is understandable given the low-level nature of POSTSCRIPT. However, it is feasible to write 'little' POSTSCRIPT programs where use is made of the graphic primitives to perform the right graphic strokes, with little effort and high gains. The red and blue books don't provide the simplest examples—they illustrate the power of POSTSCRIPT—the codes are frightening and might put you off. Maybe my backside of the envelope codes will persuade you to try some gems of your own.

But, ... the proofing is cumbersome still, alas. This can be counteracted by a discipline of POSTSCRIPT coding, hopefully[8].

2.3 The audience

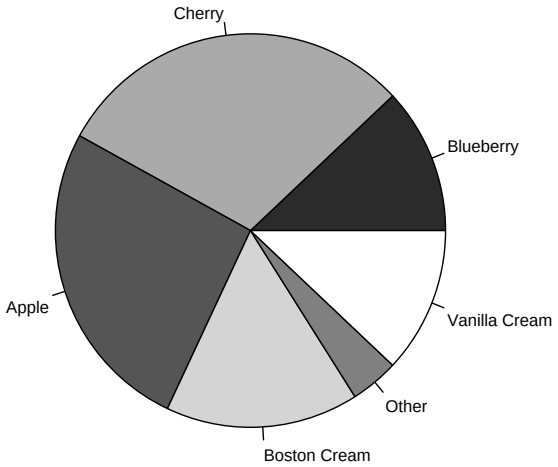
This paper is aimed at users of (La)TEX who agree with me that graphics has all to do with the right strokes. Once we know those it should be a simple coding problem to draw these strokes by POSTSCRIPT. The latter is explained along with simple examples, which are prompted by generally required functionalities.

On the other hand, as communicated by Nicolaj Tretjakov, this paper might not be that relevant, because people in practice receive POSTSCRIPT files along with some representation of a script to coerce it into a (translated) book or so. I don't know how to address that audience, as yet.

2.4 Documentation

The red book—the reference manual—is generally recommended, though the blue book—the tutorial and cookbook—is also nice.[9]

Example (Pie chart from the blue book)



The invoke is essentially as follows and shows that the codes can be used straightforwardly.[10] It is no longer necessary to mesh around with the picture environment or so, to achieve the effect.

```
%preliminary matter
(January Pie Sales)
  24 12 %... array size
  [ [(Blueberry) .12 ]
    [(Cherry) .30 ]
    [(Apple) .26 ]
    [(Boston Cream) .16 ]
    [(Other) .04 ]
    [(Vanilla Cream) .12 ]
  ] 306 396%translate center to
    140    %size
    DrawPieChart
showpage
```

[6] Courtesy Trevor J. Darrell.

[7] See also the LATEX Companion.

[8] Well, professionally there is no other way then to resort on the high-level tools.

[9] I used the POSTSCRIPT I red book and this is well-suited to get the flavour. For TEX and METAFONT this is similar. To grasp the basic ideas Knuth's first book is a more concise survey of the main lines of thought than The TEXbook and *he METAFONTbook.*

[10] It is not standard POSTSCRIPT. We have to construct some kind of library to use the PostScript programs from. Maybe the CTAN as global network library? Copied on the various CD-ROMS?

2.4.1 PostScript FAQ

There is also a POSTSCRIPT-FAQ, consult

```
ftp  wilma.cs.brown.edu:
       pub/comp.lang.postscript.
```

It contains an annotated bibliography as well.

The examples from Adobe's blue book are available on the net.

2.5 Subset 0 from the language

POSTSCRIPT is stack-oriented. This means that operations are prescribed in polish-reverse notation, also known as postfix notation, similar to the HP pocket calculators. Addition—use of operator `add`—for example is notated as follows.

```
2 3 add%yields 5 on the stack, 2 3 consumed
```

POSTSCRIPT is artificially structured via structure information in comments, double %-ed comments. Programs which obey the Adobe structure are called conforming and this is usually needed for inclusion within (La)TEX, especially the BoundingBox line is required.

Example (Conforming EPS structure)

```
%! PS EPS
%%Title: <name>
%%Creator: <name>
%%CreationData: <date>
%%BoundingBox: <llx> <lly> <urx> <ury>
%%DocumentFonts: (atend)
%%EndComments
<prolog>
%%EndProlog
%%Page: 0 1
<page 1>
%%Page: 1 2
<page 2>
%%Trailer
<...>
%%DocumentFonts: Times-Roman ...
%%Pages: 3
%%EOF
```

Creating and drawing paths is done by separate operators. For creating paths operations like `moveto` are provided while drawing goes via `stroke`.

```
0 0 moveto 0 10 lineto%create path
stroke%draw a v-line of 10pt height
```

Variables—names to be associated with their values—are handled via the so-called dictionaries. The functionality can also be obtained via procedures.

```
/size {10} def
```

The so-called literal name is preceded by a slash to distinguish the declaration from its invoke. The invoke is done by just the name, also called executable name. The procedure text is surrounded by curly braces. Parameters are absent too. The (operand) stack is used.

For graphics we have a CurrentTransformMatrix—CTM—which maps the user space on the device space, the printer or screen. Equally powerful is the concept of encapsulating graphics via `gsave` and `grestore`, that is the graphics state is local—encapsulated—after `gsave` until `grestore`.

Next to the CTM POSTSCRIPT maintains the currentpoint and currentpath.

Batagelj, MAPS 95.1—Combining TEX and POSTSCRIPT—provides an in a nutshell overview.[11] Another introduction is in Fokker en van Oostrum's 'Plaatjes in een tekst,' MAPS 94.2, next to a survey of drawing software.

2.5.1 Snapshot of (graphics) commands

The following summary is borrowed from Gurari, well ...a little modified.[12] Its main purpose is to show that the number of relevant graphic primitives is low. The functionalities will be dealt with in the examples along the way. For the details of the commands or the list of operators see the red book.[13]

Arithmetic and math operators

⟨*num*⟩ ⟨*num*⟩ mul *num*
⟨*num*⟩ sine *num*

Path construction operators

currentpoint *x y*
⟨*x*⟩ ⟨*y*⟩ moveto
⟨*dx*⟩ ⟨*dy*⟩ rmoveto
⟨*x*⟩ ⟨*y*⟩ lineto
⟨*dx*⟩ ⟨*dy*⟩ rlineto
⟨q_{1x}⟩ ⟨q_{1y}⟩ ⟨q_{2x}⟩ ⟨q_{2y}⟩ ⟨p_{2x}⟩ ⟨p_{2y}⟩ curveto
⟨c_x⟩ ⟨c_y⟩ ⟨*r*⟩ ⟨ang_1⟩ ⟨ang_2⟩ arc

String operators

⟨*string*⟩ ⟨*num*⟩ ⟨*num*⟩ getinterval

Character and font operators

/⟨*fontname*⟩ findfont
⟨*fontsize*⟩ scalefont setfont
⟨*string*⟩ show
{⟨*body*⟩} ⟨*string*⟩ kshow

Graphics state operators

⟨*num*⟩ setgray
⟨*num*⟩ setlinewidth

Dictionary operators

/⟨*defname*⟩ {⟨*body*⟩} def

Coordinate system and matrix operators

⟨*num*⟩ ⟨*num*⟩ translate

[11] Nice are the hints to remove repeated parts from files which are generated by CorelDRAW and Mathematica, in order to reduce the size of the automatically generated and to be included files. (The idea is to remove duplicate 'dictionaries' which are included with each result.) The example of how to include graphs of math functions in a document is *very* useful. However, with respect to his first picture I would prefer to use the inherent symmetry in the data as opposed to providing all the data.

[12] Gurari E.M (1994): TEX & LATEX—Drawing & Literate Programming. McGraw Hill. ISBN 0-07-025208-4.

[13] A complete list with functional summaries is in the red book Section 6.2 Operator summary.

⟨*num*⟩ ⟨*num*⟩ scale
⟨*num*⟩ rotate

Relational, boolean, and bitwise operators

⟨num_1⟩ | ⟨$string_1$⟩ ⟨num_2⟩ | ⟨$string_2$⟩ le *bool*

Control operators

⟨*bool*⟩ {⟨*truepart*⟩} {⟨*falsepart*⟩} ifelse
⟨*num*⟩ {⟨*body*⟩} repeat
⟨*from*⟩ ⟨*step*⟩ ⟨*to*⟩ {⟨*body*⟩} for

With postfix notation a sentence like the following is elegant, and close to the familiar input → output.

⟨*in*⟩ ⟨*operator*⟩ ⟨*result*⟩

2.6 What is not allowed as EPS?

I'm not knowledgeable enough to answer that question, nor do I know of a full-blown definition of EPS.[14] For the moment I consider EPS as some subset which works with all interpretors, with my subset 0 in there. When one restricts oneself to the basics of graphics, arithmetics and similar operations then the boundary between EPS and full POSTSCRIPT—or its various implementations—is not in sight.

2.7 Proofing

For previewing or printing, *as such* I have to include a shift to move the picture away from the lower left corner, say

```
300 500 translate
```

2.8 Inclusion

I usually build a figure symmetrically around the origin and then include it in my TEX document via

```
$$\psfig{file=<name>,height=<number><unit>}$$
```

A unit can be `in`(ch), `cm`, and ilks. `\psfig` is very vulnerable to spaces because of TEX's parsing. So no spaces in there. Now and then I forget to inactivate the `translate` needed while previewing. No real problem.

2.8.1 BoundingBox

Providing the right BoundingBox coordinates has all to do with proper placement within context, the look-and-feel. Default POSTSCRIPT assumes the origin—in user space—at the lower left corner of the paper—in device space.

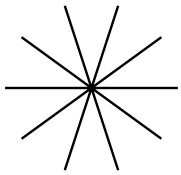
Surround the picture by as-if lines and supply the coordinates, in points as units in user space coordinates, of the lower left corner and the upper right corner in the BoundingBox specification. Simple is to build a picture around its symmetry point—and let this coincide with the origin—with as pleasing result that the horizontal positioning comes out centered, when used within math display. Vertically, I add a 10 or so extra on either side in the BoundingBox specification, but that depends on the character of the picture.

Some preview systems can measure the BoundingBox and allow adjustment interactively.[15]

2.9 Writing PostScript

A line bundle and a variant of it are introduced to show how to create simple EPS.

2.9.1 A line bundle



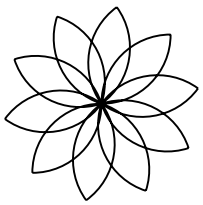
How to do this in POSTSCRIPT? A line as such is simple. First a `moveto` and then a `lineto`. So a way is to create a loop and repeatedly draw from the origin to the end of the various lines. This can be done elegantly by using appropriately the CTM.

```
%! PS EPS
%%Title: Line bundle
%%Creator: cgl
%%CreationDate: June 4 1996
%%BoundingBox: -40 -45 40 45
%%Pages: 1
%%EndProlog
%%Page: 1 1
/r 36 def
10{0 0 moveto r 0 lineto
   36 rotate
}repeat stroke showpage
```

Explanation. The idea is that first a simple line is draw, for example along the x-axes. What happens if after that we rotate? Right, the mapping is changed. And what happens if we supply the *same* line after this? Indeed, it will show up rotated. Because POSTSCRIPT is an interpretive language we can realize this specification after the rotate via a loop, which for this simple case reads `10{...}repeat`.[16]

> Appropriately maintaining the CTM for symmetrical pictures can yield simple looking POSTSCRIPT programs.

2.9.2 A flower



[14] Gurari has pointed to some information on the net but it looks informal to me.

[15] For a summary of tools to assist finding the BoundingBox coordinates see, Reckdahl K (1995): Using EPS graphics in LATEX documents. reckdahl@leland.stanford.edu or his 1996 TUGboat tutorial.

[16] Do you see the variant for drawing a polygon? This duality line bundle and polygon has been used by Gabo and is about what he called stereometry versus perimetry, the structure versus the surface

This exercises the use of `arc`.

```
%! PS EPS
%%Title: Flower
%%Creator: cgl (Courtesy Papert)
%%CreationDate: June 4 1996
%%BoundingBox: -40 -45 40 45
%%Pages: 1
%%EndProlog
%%Page: 1 1
/r 36 def
10{r r moveto%begin drawing point
   r 0 r 90 180 arc
   currentpoint%origin
   0 r r 270 360 arc
   36 rotate
}repeat stroke showpage
```

Explanation. We have the same structure as the previous program but the 'line' is now a little more elaborated: two arcs of a circle. POSTSCRIPT provides an operator for drawing circular arcs, called arc. The arc has (x, y) as centre, r as radius, ang_1 the angle of a vector from (x, y) of length r to the first endpoint of the arc, and ang_2 the angle of a vector from (x, y) of length r to the second endpoint of the arc.[17] These arguments are expected to be on the stack.

`x y r ang1 ang2 arc`

Important is to realize that arc *counts* its angle from (x, 0) and that the *drawing* starts from the point on the stack

The specification of the flower in METAFONT/MetaPost reads essentially as follows.

```
for k:= 1 upto 10:
draw(origin{up}..{right}(up+right){down}..
    {left}origin) rotated 36k;
endfor
```

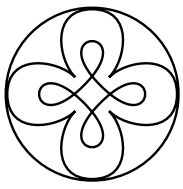
Explanantion. METAFONT allows for specification of the directions[18] $up = (0, 1), right = (1, 0)$.

IMHO, with all respect the METAFONT and POSTSCRIPT programs are similar modulo some syntactic sugar. However, the extra possibility of specifying the directions is more convenient than using control points. But perhaps that is a matter of taste, although the handling of control points is powerful as Bézier himself has shown in the past. From this I conclude that for these simple kinds of pictures we can as well use POSTSCRIPT straightaway.

3 Some more Graphics

Example (Malbork window)

This is all about using `curveto`, especially choosing suitable control points.



```
%! PS EPS
%%Title: Malbork Window
%%Creator: cgl
%%CreationDate: May 21 1996
%%BoundingBox: -40 -40 40 40
%%Pages: 1
%%EndProlog
%%Page: 1 1
45 rotate 10 0 moveto
4{20 0 37.5 12.5 25 25 curveto
  12.5 37.5 0 20 0 10 curveto
 90 rotate
}repeat%inside lops next
5 0 moveto
4{5 35 35 5 0 5 curveto
  90 rotate
}repeat%enclosing circle next
36 0 moveto
0 0 36 0 360 arc
stroke showpage
```

Explanation. `translate` changes the CTM, with the effect that the device coordinates are shifted. (Useful for use of POSTSCRIPT alone out of context.)

`rotate` changes the CTM, and because of being an interpretive language the various loop traversals map the *same* user coordinates on the rotated device coordinates.

⟨*number*⟩{...} `repeat` is a loop to be traversed ⟨*number*⟩ of times.

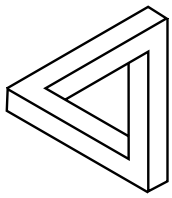
`curveto` adds a spline to the current path from the currentpoint to the last point on the stack. The first two points are the so-called control points of the spline.[19]

`arc` adds a circular arc to the current path from the currentpoint.

The details of the arguments for the operators are nicely documented in the red book.

Example (Escher's impossible triangle)

This is all about *wrong* projections. However, these kinds of pictures are intriguing and fun. I consider them well-suited to illustrate POSTSCRIPT's drawing capabilities.



```
%! PS EPS
%%Title: Escher's impossible triangle
%%Creator: cgl (inspired  by Guy Shaw)
%%CreationDate:  May 23 1996
%%BoundingBox: -40 -40 40 40
%%Pages: 1
```

[17] The arc is drawn counter clockwise. `arcn` draws clockwise.

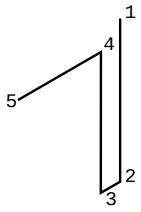
[18] There is also a quartercircle which apart from orientation is drawn similarly.

[19] Much similar as in METAFONT. Choosing for the inner lop the control points in this way is borrowed from Haralambous Y (1995): Some METAFONT techniques. TUGboat 16.1, 46–53. It is also supplied in the description of `curveto` in the red book.

```
%%EndProlog
%%Page: 1 1
3{25  34 moveto
  25 -34 lineto
  17 -38.2 lineto
  17 20 lineto
 -17.6 0 lineto
120 rotate
}repeat stroke showpage
```

5 points, the right stroke and a rotation or two, that's it. End of story.

However, it is all about finding those 5 points.




```
%! PS EPS
%%Title: Essential stroke
%%Creator: cgl (inspired  by Guy Shaw)
%%CreationDate: May 23 1996
%%BoundingBox: -40 -40 40 40
%%Pages: 1
%%EndProlog
%%Page: 0 1
25  34 moveto currentpoint
0  -68 rlineto currentpoint%down
  -120 rotate
25  34 lineto%preserve symmetry
   120 rotate currentpoint
17  20 lineto currentpoint
-17.6 0 lineto currentpoint
%labels
/Courier findfont 8 scalefont setfont
moveto -5 -3 rmoveto (5) show
moveto 1 1 rmoveto (4) show
moveto 2 -5 rmoveto (3) show
moveto 2 0 rmoveto (2) show
moveto 2 0 rmoveto (1) show
stroke showpage
```

Explanation. The essential stroke figure also illustrates the integration of text in this case digits. `currentpoint` pushes the point on the stack. The last `moveto`-s pop these coordinates up. `rmoveto` moves *relatively.*

And what about their relationships, and what about the minimal information to be prescribed?

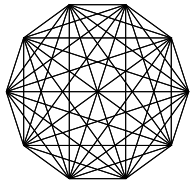
Looking more closely it turns out that *only* the first point is all that is needed. The rest is implicit to the nature of the figure.[20]

```
%! PS EPS
%%Title: Escher's Impossible triangle II
%%Creator: cgl
%%CreationDate: May 23 1996
%%BoundingBox: -40 -40 40 40
%%Pages: 1
%%EndProlog
%%Page: 1 1
%Parameterized over p1
/point {25 34} def%note x<y
%
3{point moveto
currentpoint neg lineto%down
  -120 rotate
point  lineto%preserve symmetry
   120 rotate
currentpoint 2 div neg lineto
currentpoint 3 sqrt mul sub 0 lineto
120 rotate
}repeat stroke showpage
```

Explanation. `currentpoint` yields the coordinates of the current point of the path on the stack. The other operations do what their names suggest. The temporarily change of the CTM within the loop expresses the rotation symmetry relation between points 1 and 3.

Example (Bentley's polygon)

This code is all about a double loop and using the loop variable from the stack, next to using the `gsave` and `grestore` advantageously.

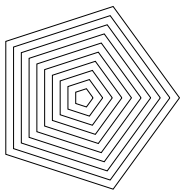


```
%! PS EPS
%%Title: Bentley's double loop
%%Creator: cgl
%%CreationDate: May 30 1996
%%BoundingBox: -100 -105  100 105
%%EndProlog
10{1 1 9{100 0 moveto
         gsave
         36 mul rotate%loopcount*36
         100 0 lineto stroke
         grestore
   } for
   36 rotate
}repeat showpage
```

Explanantion. `gsave` and `grestore` are needed to draw locally, that is at the end the graphics state—currentpoint, currentpath and CTM—is restored with the values at the beginning. 1 1 9 stand for beginvalue step and endvalue of the `for` counter.

Example (Another double loop)

A set of nested polygons provide also a double loop situation.



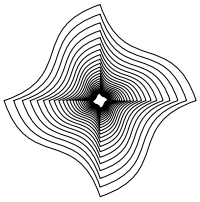
```
%! PS EPS Nested pentagons
%%Title: Pentagons
%%Creator: cgl
%%CreationDate: June 17 1996
%%BoundingBox: -100 -100 100 100
%%Pages: 1
```

[20] Of course one can also think of other equivalent parameters like size and thickness.

```
%%EndProlog
%%Page: 1 1
10 10 100{dup 0 moveto
   5{72 rotate
     dup 0 lineto
     }repeat
}for stroke showpage
```

Example (Polygons with splines as sides)

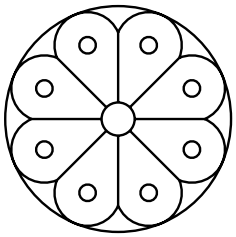
This generalization of polygons was introduced by Jackowski at EuroTEX 95. A special case of METAFONT's interpath functionality is shown en-passant.



```
%! PS EPS Nested 'squares'
%%Title: polygon.eps II
%%Creator: cgl
%%CreationDate: June 17 1996
%%BoundingBox: -100 -100 100 100
%%Pages: 1
%%EndProlog
%%Page: 1 1
/r 100 def
/r1 {r .25 mul} def
/r3 {r .75 mul} def
25{r 0 moveto
   4{r3 r3 r1 r1 0 r curveto
     90 rotate
     }repeat
   .9 .9 scale
}repeat stroke showpage
```

Example (Barn window)

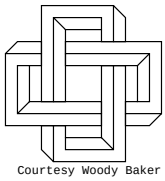
This is all about playing with circles and circular arcs.[21]



```
%! PS EPS
%%Title: Barn Window II
%%Creator: cgl
%%CreationDate: May 29 1996
%%BoundingBox: -45 -45 45 45
%%Pages: 1
%%EndProlog
%%Page: 1 1
/l 36 def
/r {l 22.5 sin mul} def
/m {l 22.5 cos mul} def
8{r .5 mul 0  moveto
  l 0 lineto
  currentpoint   %begin circular arc
  22.5 rotate m 0%center
  r              %radius
  -90 90 arc
  22.5 rotate
}repeat
%inner circle
/rin {r .5 mul} def
rin 0  moveto
0 0 rin 0 360 arc
%outer circle
/rout {r m add} def
rout 0  moveto
0 0 rout 0 360 arc
%extra circles
/rin {r .25 mul} def
22.5 rotate
8{m rin add 0 moveto
  m 0 rin 0 360 arc
  45 rotate
}repeat stroke showpage
```

I'm sure I'll come back some day and look again through this window, but then pastel colored.

Example (Baker's inspiration)

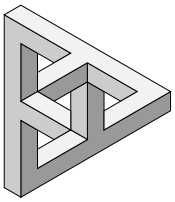


This example is similar to Escher's impossible triangle. Find the essential stroke and rotate.

```
%! PS EPS
%%Creator: cgl (inspired  by Woody Baker)
%%CreationDate: May 1996
%%BoundingBox: -80 -80 80 80
%%Pages: 1
%%EndProlog
%%Page: 1 1
4{-15 25 moveto
    0 -10 rlineto
   60 0 rlineto
    0 -30 rlineto
   10 0 rlineto
    0 40 rlineto
  -70 0 rlineto
    0 10 rlineto
   80 0 rlineto
    0 -60 rlineto
  -30 0 rlineto
    0 10 rlineto
   10 0 rlineto
%
   35 -25 moveto
    0 -10 rlineto
   20 0 rlineto
   10 10 rlineto
%
   45 15 moveto
   10 10 rlineto
90 rotate
}repeat stroke
%
/Courier findfont 10 scalefont setfont
-55 -75 moveto
(Courtesy Woody Baker)show
showpage
```

[21] The first example in the blue book collection provides a similar picture with gradually changing scales of grey.

Example (Romanovsky's real Escher)

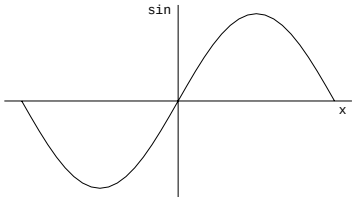


Grey scales can be obtained simply via ⟨*number*⟩ `setgray`, with ⟨*number*⟩ ∈ [0, 1]. 0 denotes black and 1 is white. The idea is to construct the essential path—the stroke denoted by a grey scale—and to use this 3 times.

4 Math graphs

In (La)TEX documents it is a problem[22] how to include accurate graphs of mathematical functions. Because of POSTSCRIPT's arithmetic and graphics capabilities it is handy to use POSTSCRIPT.

Example (Sine function)




```
%! PS EPS
%%Title: Sine function
%%Creator: cgl (inspired  by Batagelj)
%%CreationDate: May 27 1996
%%BoundingBox: -200 -110  200 110
%%EndProlog
/Courier findfont 15 scalefont setfont
%x-axes and label
-200 0 moveto 200 0 lineto
-15 -15 rmoveto (x) show
%y-axes and label
0 -110 moveto 0 110 lineto
-35 -10 rmoveto (sin) show
%function
-180 0 moveto
-180 10 180{%from step to
   dup sin 100 mul%(x, 100sin x)
   lineto
}for stroke showpage
```

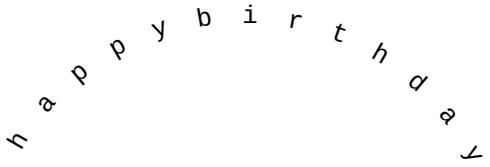
The invoke might read as follows.

```
$$\psfig{file=sine.eps,height=1in}$$
```

5 Text set along curved paths

A teaser. With the advent of scalable and rotationable outline fonts this is possible too.[23]

Example (Along a circle)




```
%! PS EPS
%%Title: Typesetting along arcs
%%Creator: cgl
%%CreationDate: June 4 1996
%%BoundingBox: -100 50 100 125
%%Pages: 1
%%EndProlog
%%Page: 1 1
/Courier findfont 10 scalefont setfont
/text (happybirthday) def
50 rotate
0 1 12{0 100 moveto
  text exch 1 getinterval show
  -10 rotate
}for stroke showpage
```

Joseph Romanovsky communicated that `kshow`—kerning (and more general positioning) under user control—is available which allows a general def to be executed between two characters of a string.

Example (Along a spiral)

The blue book provides an example of typesetting along a path—a quotation of Woody Allen—where the path accentuates his filmmaker profession. The example below shows a nice effect with little knowledge of POSTSCRIPT, essentially the use of `kshow`.


```
%! PS EPS
%%Title: Text along spiral
%%Creator: J.V. Romanovsky
%%CreationDate: Adapted from JVR June 96
%%BoundingBox: -100 -90 60 70
%%EndProlog
/Courier findfont 20 scalefont setfont
-100 0 moveto 50 rotate
{-10 rotate 3 0 rmoveto .98 .98 scale}
 (Olga Grineva my charming
  St Peterburg hostess)kshow
showpage
```

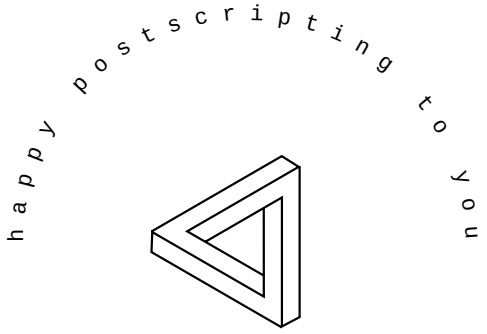
[22] Communicated by Nico Temme. He solved the problem by doing the calculations in PASCAL. For advanced manipulations Mathematica or Mapple are generally used where the resulting EPS is pasted up in the (La)TEX script as usual.

[23] Disclaimer: Typesetting math along curved paths is something different.

Example (Seals)

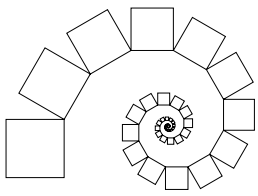
The problem has been discussed by Hoenig at EuroTEX 92, and Zlatuška at EuroTEX 95, both biased by METAFONT. POSTSCRIPT alone is suited too with an overall simpler process. Combining two earlier supplied examples yields Zlatuška's seal in principle.[24]

```
%! PS EPS
%%Title: Seal, in principle
%%Creator: cgl
%%CreationDate: June 6 1996
%%BoundingBox: -110 -45 110 100
%%Pages: 1
%%EndProlog
%%Page: 1 1
%150 650 translate
/Courier findfont 10 scalefont setfont
/text (happy postscripting to you) def
/r 100 def
gsave
   90 rotate %begin orientation
   0 r moveto%begin point
{-7.04 rotate 0 r moveto} text kshow
grestore%next the central Escher
3{25  34 moveto
  25 -34 lineto
  17 -38.2 lineto
  17 20 lineto
 -17.6 0 lineto
120 rotate
}repeat stroke showpage
```



Remark. The difference of fonts in the main text and that used by POSTSCRIPT is no longer there when POSTSCRIPT fonts are used throughout, be it the POSTSCRIPT version of the CM family.

Example (Gurari's squares)



```
%! PS Gurari squares
%%BoundingBox: -200 -110  200 110
%%Creator: cgl
%%CreationDate: June 20 1996
%%EndProlog
/r 22 def
/square {1 1 4{0 r rlineto
         90 rotate}for} def
0 0 moveto 90 rotate
50{r 0 rmoveto square
   -30 rotate .9 .9 scale
}repeat stroke showpage
```

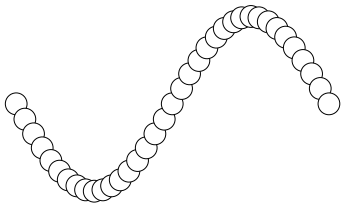
Example (Gurari's ABC)

Very nice this suggestion of motion.


```
%\PS EPS
%%Title: Gurari's ABC
%%BoundingBox: 0 -45  100 75
%%Creator: Gurari
%%CreationDate: copied June 17 1996
%%Pages: 1
%%EndComments
%%EndProlog
%%Page: 1 1
/Times-Bold findfont 45 scalefont setfont
-40 rotate
1 -.03 0{setgray
         0 0 moveto
         (ABC) show
         3 rotate
        } for
0 0 moveto -4 rotate
1 setgray (ABC) show
showpage
```

Example (Walking along the S-curve)

In *The METAFONTbook* ex13.10 is about drawing overlapping disks along a path, the S-figure. How to do this in POSTSCRIPT straightaway? There is no 'point of' a path operator so the best we can attain is to walk along a math function.[25]



```
%\PS Sine with overlapping disks
%%BoundingBox: -200 -110  200 110
%%Creator: cgl (METAFONTbook ex13.10)
%%CreationDate: June 17 1996
%%EndProlog
newpath
-180 10 180{dup sin 100 mul%(x, 100sin x)
```

[24]The blue book also provides a seal—Symphony No.9—but that is more complex and ipso facto requires more knowledge of POSTSCRIPT to understand what is going on, IMHO, with all respect. To set this poster from the blue book is no more difficult than the use of `arc`, however. My example gives you the feeling that you understand what is going on.

[25]Or specify a path explicitly of course.

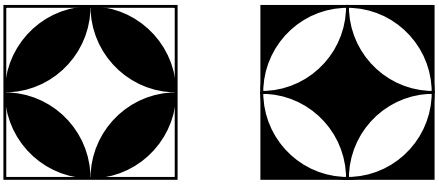
```
    12.5 0 360 arc
    gsave 1 setgray fill grestore
    stroke
}for showpage
```

Explanation. `1 setgray fill` is the erase functionality, encapsulated to yield what we want.

6 Reverse video

Let us go back to the flower picture as given at the beginning of this note and reuse the leaves.

Example (reverse video)



The above is obtained via

```
%\PS Reverse video
%%BoundingBox: -30 -35 120 35
%%Creator: cgl
%%CreationDate: Aug 1996
%%Pages: 1
%%EndProlog
%%Page: 1 1
/r 30 def
/filling {fill} def
/tile {4{r 0 moveto
   0 0 r 0 90 arc
   currentpoint
   r r r 180 270 arc
   filling
   90 rotate
   }repeat
} def
/frame {r neg r moveto
 r 2 mul 0 rlineto
 0 r -2 mul rlineto
 r -2 mul 0 rlineto
 closepath
} def
%
%tile
%
gsave
 tile frame stroke
grestore
%
%reverse video tile
%
r 3 mul 0 translate
frame
gsave fill grestore%background
stroke
/filling{gsave 1 setgray fill
         grestore}def
tile stroke showpage
```

Reverse video in POSTSCRIPT can be obtained via providing a black background and for the picture replace `fill` by `gsave 1 setgray fill grestore`.

7 Tiling

Tiling is all about copies of an element, shifted and/or rotated, to fill up space traditionally in the plane. Below a leave is 'copied' four times and the resulting tile is 'copied' four times. The copying comes down to redoing the figure at the prescribed place eventually rotated. The latter is possible by modifying the CTM via `translate` or `rotate`.

Example (Tiling)

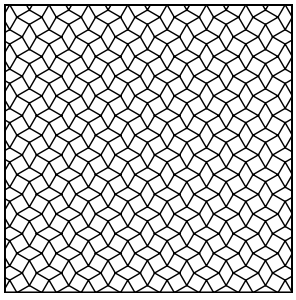
The above is obtained as follows.

```
%\PS Tiling and reverse video
%%BoundingBox: -45 -40 40 45
%%Creator: cgl
%%CreationDate: Aug 1996
%%Pages: 1
%%EndProlog
%%Page: 1 1
/r 20 def
/tile {4{r 0 moveto
 0 0 r 0 90 arc
 currentpoint
 r r r 180 270 arc
 fill
 90 rotate
 }repeat
}def
/frame {r neg r moveto
 r 4 mul 0 rlineto
 0 r -4 mul rlineto
 r -4 mul 0 rlineto
 closepath
} def
%
frame stroke%no clipping necessary
2{2{tile r 2 mul 0 translate
    }repeat r -4 mul r -2 mul translate
 }repeat
r r neg translate
showpage
```

8 Clipping

The clipping functionality is different in spirit from METAFONT. In POSTSCRIPT we have to adjust the frame to draw within via creating a path and clip this path, that is make this path the drawing boundary, that is all.

Example (Clipping)



The above is obtained via

```
%\PS Tiling fourthree
%%BoundingBox: -100 -100 100 100
%%Creator: cgl
```

```
%%CreationDate: Aug 1996
%%Pages: 1
%%EndProlog
%%Page: 1 1
%300 500 translate
/a 10 def
/ha {a .5 mul} def
/tile {% rhombus + 90 rotated rhombus
 ha 3 sqrt mul 0 moveto
 0 ha lineto
 ha 3 sqrt mul neg 0 lineto
 0 ha neg lineto
 closepath
 ha 1 3 sqrt add mul ha 3 sqrt mul lineto
 ha 2 3 sqrt add mul 0 lineto
 ha 1 3 sqrt add mul ha 3 sqrt mul neg lineto
 closepath
} def
/tena {a 10 mul}def
/frame {tena neg tena moveto
 tena 2 mul 0 rlineto
 0 tena -2 mul rlineto
 tena -2 mul 0 rlineto
 closepath
} def
/dotiling {
 a -11 mul tena neg translate
 9{gsave
  11{tile a 1 3 sqrt add mul 0 translate
    }repeat stroke
  grestore
  gsave ha 1 3 sqrt add mul dup translate
  11{tile a 1 3 sqrt add mul 0 translate
     }repeat stroke grestore
  0 a 1 3 sqrt add mul translate
 }repeat
} def
%
%tile
%
frame clip dotiling showpage
```

9 Tables set sideways

Another teaser is to set tables rotated. Rokicki provided rotate macros along with his `dvips`, among others. I have borrowed the essence from his rotate macros and recast into the following.

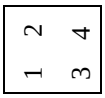
```
\def\rotate#1%stuff
          #2%degrees in PS direction
{\setbox\abox=\hbox{#1}%
 \adim\ht\abox\advance\adim by\dp\abox
 \hbox to\adim{\vbox to\wd\abox
 {\vskip\wd\abox
 \special{ps: gsave
    currentpoint currentpoint translate
    #2 neg rotate
    neg exch neg exch translate}%
 \box\abox\vss}\hss}%
 \special{ps: currentpoint
       grestore moveto}%
}%end rotate
```

The point is that it is not much. The `ps:` is dependent on the system still, alas.

Example (Rotated table)

The example works under UNIX with Rokicky's `dvips`.

```
\def\data{1\cs2\rs
          3\cs4 }
pre
\rotate{\framed
        \btable\data}
       {90}
post
```

pre 

post

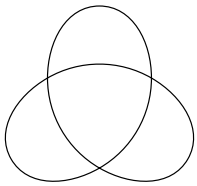
Remark. Gurari (1994) has provided some more examples of POSTSCRIPT↔(Al)DraTEX interaction. Real interfacing. Apart from portability problems it gets quite complicated. For the moment I refrain and code the 'pictures' in raw POSTSCRIPT assisted by METAFONT for prompting control points of those curves which can be specified elegantly in a declarative way. And, of course, there is the wealth of PStricks.

10 METAFONT/MetaPost user interface

Sometimes it is more natural to specify points and *directions*. It is true that specifying a control point along the direction can yield the same effect but the distance between the point and its control point influences the shape.

Example (Escher's knot)

This example is all about specifying directions.



In METAFONT the coding would read as follows.

```
%Escher's Knot. June 96. cgl@rc.service.rug.nl
def openit = openwindow currentwindow
   from origin to (screen_rows,screen_cols)
   at (-2r,3r)enddef;
pickup pencircle scaled 1;
tracingstats:=proofing:=1; screenstrokes;
pair p[];
%parameters
r:=100; alfa=90;
%
p2:=(0,.85r); %independent from p1,3,4
p4:=(0,-.5r);
%dependent points because of symmetry
p1:=p4 rotated -120;
p3:=p4 rotated  120;
path q;
q=p1{dir alfa}..{(1,0)}p2..
    {dir(-alfa)}p3..{dir(alfa-240)}p4;
draw q;
draw q rotated 120;
draw q rotated-120;
showit;
end
```

By the nature of the figure not only points are related but also their directions. How to cope with this in POSTSCRIPT? It can be done but not so elegantly, honestly speaking it is quite cumbersome. But ...there is a solution or two, hang on.

```
%! PS EPS
```

```
%%Title: Escher knot II
%%Creator: cgl (inspired  by Knotplot)
%%CreationDate: June 1996
%%BoundingBox: -95 -95 95 95
%%Pages: 1
%%EndProlog
%%Page: 1 1
%
/angle 90 def
/r 100 def
/point {0 -.5 r mul}def
/p1 {-.25 r mul 3 sqrt mul  .25 r mul moveto
     currentpoint
     angle sin 2 mul add exch
     angle cos 2 mul add exch
     -20 .85 r mul
     0 .85 r mul
     curveto stroke} def
/p3 { .25 r mul 3 sqrt mul  .25 r mul moveto
     %Control point
     currentpoint
     angle sin -15 mul add  exch
     angle cos  15 mul add  exch
     %Control point:
     % 58.62 -.5 r mul 5 add%angle 90
       0       angle -240 add cos -15 mul add
     -.5 r mul angle -240 add sin -15 mul add
     0 -.5 r mul
     curveto stroke} def
3{p1
  gsave -1 1 scale p1 grestore%reflect
  p3
  120 rotate
}repeat showpage
```

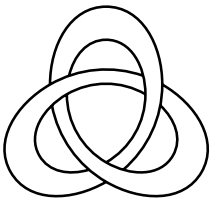
Explanation. The essential curve is split into 3 pieces: $p_1,\ldots p_3$. The first two are related by reflection. The third must properly match. In the Columbus's egg paragraph the straight `.eps` code is given, biased by the knowledge of the (control) points.

10.1 A teaser

More complicated is when the line is changed into a tube, and when we have to deal with hidden lines. In METAFONT the code could read as follows, where use is made of `intersectiontimes` and of subpaths.[26]

Example (Escher's doughnut)

This example is all about getting from a *declarative* specification in METAFONT to an *imperative* EPS code.



First the declarative METAFONT code.

```
%Escher Knot III. June 96.
% cgl@rc.service.rug.nl
def openit = openwindow currentwindow
   from origin to (screen_rows,screen_cols)
   at (-2r,3r)enddef;
pickup pencircle scaled 1;
tracingstats:=proofing:=1; screenstrokes;
numeric t, u, v, w;
pair p[]; path q[];
def assignpoints=
 p2:=(0,.85r); %independent from p1,3,4
 p4:=(0,-.5r);
 %dependent points because of symmetry
 p1:=p4 rotated -120;
 p3:=p4 rotated  120;
enddef;
%
alfa=90;r:=100;  assignpoints;
q1:=p1{dir alfa}..{(1,0)}p2..
    {dir(-alfa)}p3..{dir(alfa-240)}p4;
%inside
        r:=.75r; assignpoints;
q2:=p1{dir alfa}..{(1,0)}p2..
 {dir(-alfa)}p3..{dir(alfa-240)}p4;
(t,u)= subpath (2.5,3) of q1
   intersectiontimes (q2 rotated -120);
(v,w)= q2 intersectiontimes
                    (q1 rotated 120);
%showvariable t,u;
draw subpath (0,2.5 + t/2) of q1;
draw subpath (0,2.5 + t/2) of q1
                           rotated 120;
draw subpath (0,2.5 + t/2) of q1
                           rotated-120;
%showvariable v,w;
draw subpath (v,3) of q2;
draw subpath (v,3) of q2 rotated 120;
draw subpath (v,3) of q2 rotated-120;
showit;
end
```

To give the reader an impression of what MetaPost will yield `escherknotIII.eps`, the imperative code, is included.[27]

```
%! PS EPS
%%BoundingBox: -40 -31 40 43
%%Creator: MetaPost and JJW, cgl
%%CreationDate: June 17 1996
%%Pages: 1
%%EndProlog
%%Page: 1 1
3{-21.6507 12.5 moveto
-21.6507 27.74551 -13.78212 42.5003
    0 42.5003 curveto
13.78212 42.5003 21.6507 27.74551
   21.6507 12.5 curveto
21.6507 -0.24506 16.04897 -12.19249
   6.58395 -20.3313 curveto
%
-14.3152 15.86746 moveto
-12.43301 23.58928 -7.45113 29.75021
   0 29.75021 curveto
9.64748 29.75021 15.15549 19.42186
   15.15549 8.75 curveto
15.15549 -2.07904 9.37823 -12.08548
   0 -17.5 curveto
120 rotate
}repeat stroke showpage
```

As can be seen from the last code it is all about finding the right (control) points and draw the strokes, as remarked at the beginning of this note. The difference between the declarative METAFONT specification and the

[26] This code works as such on my Mac with Bluesky's PD METAFONT. For other environments build a character from it, or adapt it for use in MetaPost, or even simpler copy the bread-and-butter EPS code which is appended at the end.

[27] A white lie. I have edited the file and reduced the data—and deleted `dtransform`, `idtransform` and the various `set...`—for the 6 strokes into only 2 and rotated these. METAFONT allowed me to declaratively specify the picture while MetaPost provided me with the essential path data. Well... even METAFONT can be asked to provide those (control) points.

resulting (unedited) MetaPost code is striking. When the last code is shown first one would say, ah... POSTSCRIPT is easy just data and some strokes. The resulting code is equivalent to Woody Baker's code: just the right stroke and a rotation or two.

KnotPlot on the net provides a more complicated version where the light reflection is emulated by shades of grey. The gzipped file is 64KB, however. A world of difference.

10.2 Columbus' egg

Why not use METAFONT to create 'the (control) points' from the descriptive picture and use these in raw POSTSCRIPT straightaway?

After assigning `precontrol`-s and `postcontrol`-s to pairs and inserting `show`-s, METAFONT yielded for the simple Escher knot the data in the transcript file. A little editing of this log file resulted in the following imperative EPS code.

```
%! PS EPS
%%Title: Escher knot (mf prompted)
%%Creator: cgl
%%CreationDate: June 1996
%%BoundingBox: -80 -80 80 80
%%Pages: 1
%%EndProlog
%%Page: 1 1
3{-43.30139 25 moveto
 -43.30139 55.49103 -27.56424 85.00061
  0 85.00061 curveto
  27.56424 85.00061 43.30139 55.49103
  43.30139 25 curveto
  43.30139 -5.94014 26.79497 -34.5299
  0 -50 curveto
120 rotate
}repeat stroke showpage
```

I presume the functionality is similar to Jackowski's `mftoeps`. The above method is my Poor Man's METAFONT2EPS, with concise, very concise and intelligible EPS as result.[28]

11 Acknowledgements

First of all Don Knuth and John Hobby thank you.

Thank you Joseph Romanovsky for showing by example the power of POSTSCRIPT, and for your cooperation on the METAFONT↔POSTSCRIPT duality.

Thank you Bogusław Jackowski for your 'POSTSCRIPT for TEXies' at BachoTEX 96, suggesting that POSTSCRIPT as such is beneficial for TEXies, next to your mftoeps.

Thank you Eitan Gurari and anonymous TEXies from whom I borrowed material, not in the least Adobe for providing POSTSCRIPT to start with.

Piet Tutelaers provided me with a copy of the PSFAQ, and Erik Frambach traced the file with examples from the blue book, next to KnotPlot.

As usual Jos Winnink proofed the paper and lend a helping hand in procrusting towards MAPS inclusion if not for processing `escherknot.mf` into `escherknot.eps` via MetaPost, although I could have done without as discussed.

Finally, thank you Erik and Wietse for the various discussions about TEX and METAFONT.

12 Conclusion

To code symmetrical and simple curves in raw POSTSCRIPT is fun and yields elegant scripts and concise files. To merge text with graphics is fun too, and the teaser to set along curved paths can be done by POSTSCRIPT elegantly. Another teaser of drawing math curves accurately along with (La)TEX is solved also by means of POSTSCRIPT. Powerful too is to *extend* the inclusion of .eps files at the dvi level by a little more interaction between (La)TEX and POSTSCRIPT. Rotating a box, with as applications for example typesetting tables in landscape, is possible in POSTSCRIPT, at the expense of system dependency because of the `\special`-s.

Merging a little knowledge of POSTSCRIPT with TEXpertise is powerful. PStricks concentrates on *interfacing* (La)TEX with POSTSCRIPT at the expense of burdening (La)TEX too much, IMHO, with all respect. For `\rotate` I interfaced too. However, for typesetting along curved paths I would not think of interfacing via rotated boxes or so.

Of course, people who do need advanced features or have special wishes might better use Adobe Illustrator, CorelDRAW, Mathematica, or Adobe Photoshop, and not to forget the pleasing MetaPost.

To understand and learn TEX did take me a couple of years. To acquaint myself with METAFONT did cost me a few months. Learning just a little bit of POSTSCRIPT was a matter of weeks, and when concentrating on paths and (control) points the `moveto`, `arc` and `curveto` can be grasped on a late afternoon.

Although the blue book contains also examples of typesetting text, I consider TEX unsurpassed for this. The best of both worlds is to combine (La)TEX and POSTSCRIPT.

Maybe we should follow Adobe and extend the use of POSTSCRIPT by PDF—or use the alternative HTML—to facilitate WWW surfing.

13 What more?

For pictures I use a TEX controlled database with the beneficial side-effect that I don't have to worry about file systems when using pictures (tools, references and ilks) on different machines. I would welcome a similar functionality for my collection of POSTSCRIPT pictures to be used with `\psfig`. I hope that the examples of the blue book next to my examples as included here—and those to

[28] Note that the final digits are 'noise.'

come—will contribute to the `.eps` library[29] for reuse or inspiration.

My anthology of examples in METAFONT will emerge in a series of notes with occasionally POSTSCRIPT alternative (hand) codings added. The first note in the series is about tiling.

A next step is the manipulation of colors either via MetaPost or POSTSCRIPT directly. Jackowski uses Adobe Illustrator for example to enrich interactively the systematic EPS pictures created by METAFONT. Indeed interesting, very interesting, but beyond my possibilities for some time to come. Neither do I have access to color POSTSCRIPT printers as yet, alas. My case rests.

Have fun, and all the best.

[29] This library was coined by Jackowski and Ryćko at EuroTEX 94 to start with their 'expanded stroke', 'removing overlap,' and 'updateB(ounding)B(ox)' if not for their mf2eps package.

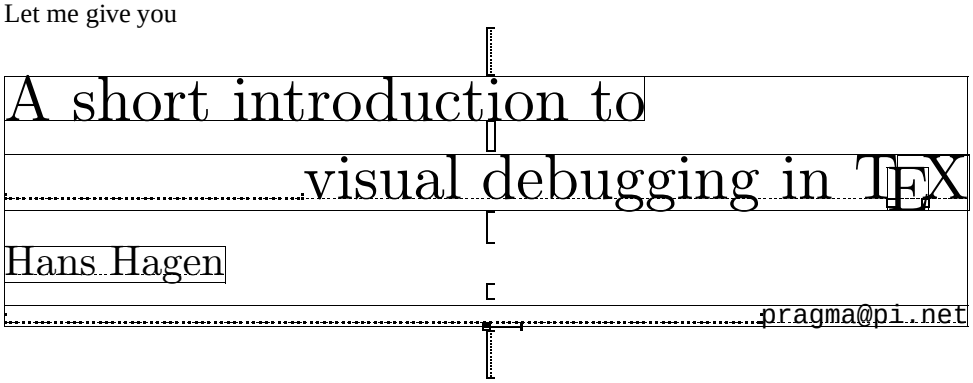
Visual Debugging in TEX

a short introduction

Hans Hagen

September 25 1996

Let me give you



This kind of fancy heading shows some dotted lines, rules and peculiar visual symbols. A more close observation learns that in fact it is some endoscopic view in what is often called TEX's stomach. For those readers who have planned to skip the rest of this article, here is how the magic is done:

```
\input supp-vis \showmakeup
```

For those who want to take a closer look at all those kerns, skips and penalties, this articles can be of some help. Although this kind of stuff often attracts the more hacking type of reader, the module described here can be of great help and provide a lot of fun to all TEX users, whatever macropackage they use.

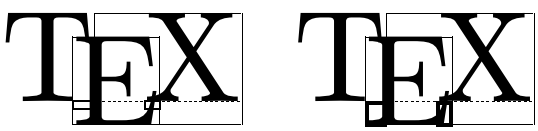
When TEX builds paragraphs and pages, it takes a lot into account. Even after years of writing macros the interference of skips, kerns, penalties, boxes and rules sometimes surprises me. One must always be aware of interline skips, top of page skips, good breaks and no breaks, either user supplied or system generated.

The idea to build some visualization macros was born while I was documenting the source of CONTEXT. Because this package is quite complete, the full documentation will be laid down in thousands of pages. Such technical documentation cannot go without showing how things are done. Because most macros at the user level have some visual impact, I decided to build a visualization tool. After having written this bunch of macros, their second purpose soon became visual debugging.

The concept is rather simple: replace the primitives `\.box`, `\.skip`, `\kern`, `\penalty`, `\.glue`, `\.ss`, `\.fil.` and `\.fil.neg` by macros that makes them visible. Most advanced TEX tutorials give examples of adapting the primitive `\par`, but somehow tampering with other TEX primitives is considered more tricky. Although the name primitive suggest that they are somehow fixed, even primitives can be `\let`'d or `\def`'d to something else. Temporary superseding the `\font` primitive is for instance needed when one wants to postpone loading of fonts in Plain TEX.

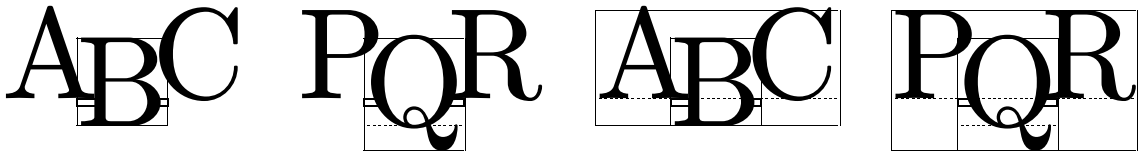
One can imagine that replacing `\hbox` with something else can have disastrous consequences. Primitives like `\setbox` expect a box and setting `\hbox` to `\relax` will surely lead to loud complaints. Some first experiments showed however that substitution was surprisingly easy. More time was spent on finding a sort of replacement that does not conflict visually when more primitives are given in a row.

Let's start with a well known piece of text. We've blown it up a bit, so we can see what happens.

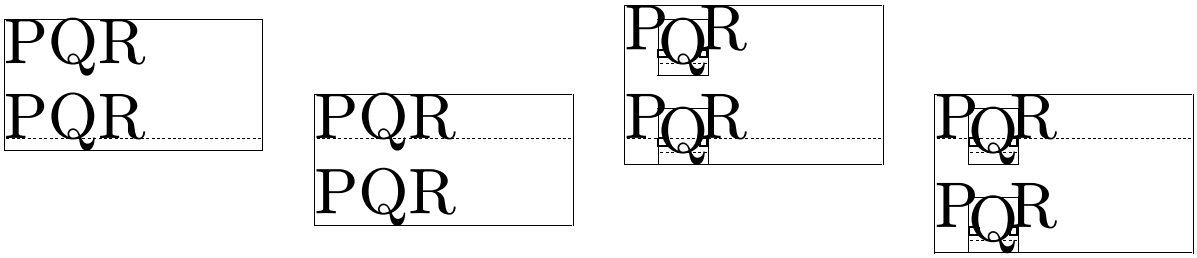


Here we see a T, followed by a kern, a boxed E, another kern and a X. The kerns have a negative sign and are visualized as small rectangles. Negative values are drawn left of their insertion point. The second TEX has exaggerated cues.

The three uppercase characters that make up TEX have no descenders. The next example shows a few more TEXed characters. This time we've got them boxed, so we can see what happens to the baseline of this combination of characters. Lowering the B and Q does not influence the baseline, which is what we expect.



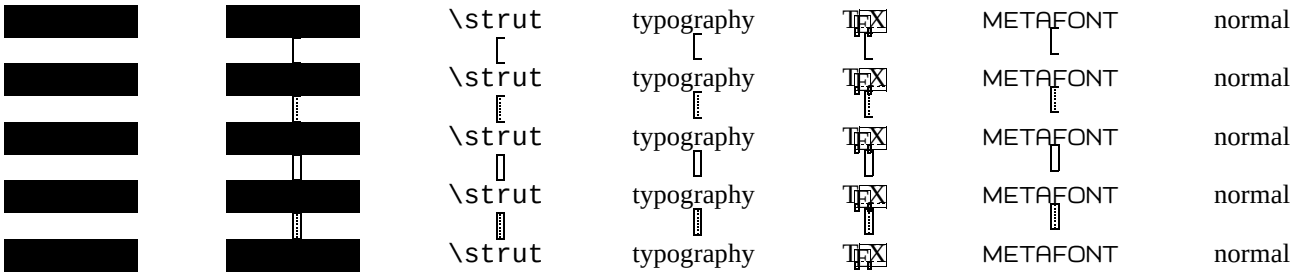
Vertical boxes come in two flavors. The default vertical box `\vbox` inherits it's baseline from the last line, while `\vtop` takes the baseline of the first line.



Visualization of fills is no problem either. In the centered line shown below the piece of text has some `\hfil`'s around it.

The same one, showing the surrounding box and two `\hfil`'s at the left of the text, looks like:

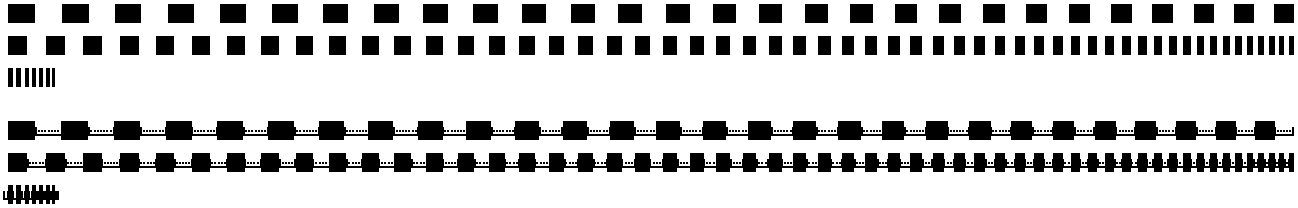
When using substitutes for the primitives mentioned, keeping the spacing intact is not always trivial. Especially the vertical spacing is very sensitive to interference. The next examples show us that at least normal situations can be handled well.



Here we see some positive vertical cues. Their negative counterparts are drawn left of the axis. Top down we see a skip, another skip with some stretch, a kern and some glue. A penalty of 100 looks like this and can be negative too. Skips, kerns and glue, which by the way is a Plain TEX macro and not a primitive, are shown at their natural size. Penalties are drawn in ranges, which are tuned to the most common cases. Combinations of penalties show up all right as we can see in where we have inserted penalties of 10000, 100 and 1.

Horizontal spacing is less sensitive than vertical spacing. Here we don't have to take interline spacing and previous depths into account. Just to prove that things work, we show a similar example here. As a bonus we've added `\hss`.

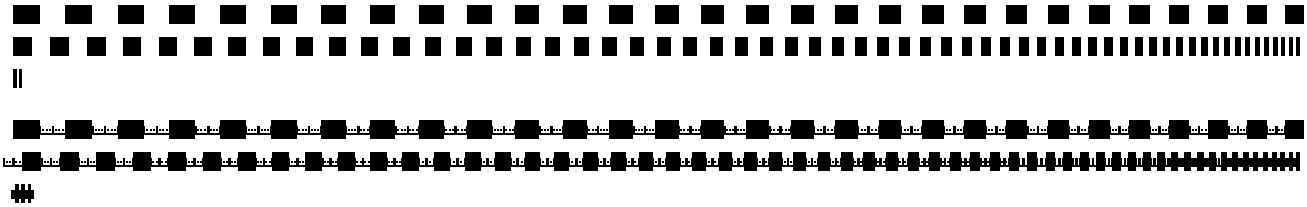
When we are typesetting in horizontal mode we have to preserve linebreaking. The next example shows a dummy paragraph with skips.



In this example it's hard to see that the stretch is equally distributed around the skip. The next line of text shows this feature in full glory. This feature is disabled by default.

hello big big world

Now look what happens when we combine two horizontal skips. This time TEX is not able to remove the visual cues. A similar situation occurs at a pagebreak. This kind of tricky situations can only be solved by an invisible kind of box, which is unfortunately not part of TEX. Of course we can backtrack skips, kerns and penalties, but such a, still not perfect, solution only complicates the macros beyond understanding.



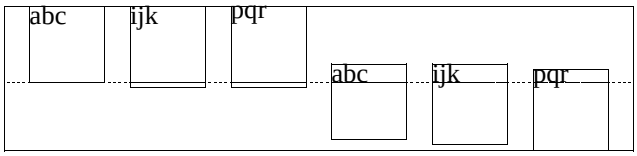
Mathematical spacing is implemented too, but due to the font-bound character, its visualization is the least impressive: $x \sqcap y$ and $x \sqcup y$ for math kern and math skip of 7 mu.

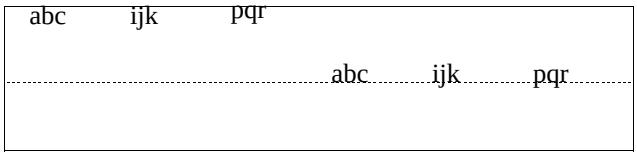
The next set of examples shows how vertical boxes are aligned when pasted together in a horizontal box. When I was messing around a bit with these samples, I became aware of some side effects that normally go unnoticed probably because they are quite natural. Confronted with these effects, I first thought that the visualization macros were somehow responsible, but additional testing proved otherwise. Of course one can never be sure, but rereading some paragraphs in Victor Eijkhouts TEX by Topic learned me that indeed such effects occur.

The samples are built up in the following way. Here the dots stand for some trailing text and/or macros.

```
\hbox to \hsize
  {\hss
   \vbox to 1cm{\hsize.15\hsize abc\par ... }\hss
   \vbox to 1cm{\hsize.15\hsize ijk\par ... }\hss
   \vbox to 1cm{\hsize.15\hsize pqr\par ... }\hss
   \vtop to 1cm{\hsize.15\hsize abc\par ... }\hss
   \vtop to 1cm{\hsize.15\hsize ijk\par ... }\hss
   \vtop to 1cm{\hsize.15\hsize pqr\par ... }\hss}
```

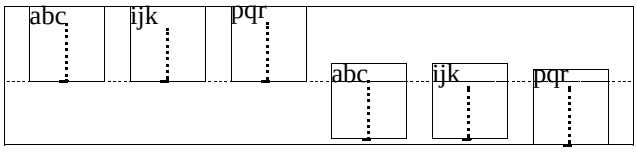
We show both the visualized example and the natural one. The latter illustrates compatibility. When we insert nothing, this pack of boxes looks like:

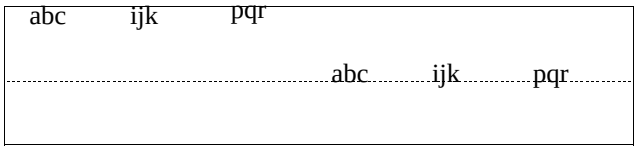




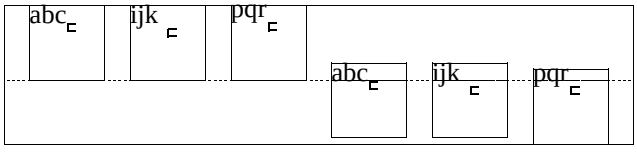
The first box has the height we expect. The second and third box also has the desired height, but here the depth of the j and q has migrated to the surrounding box. The height and depth of the fourth box totals to 1 cm, and we don't recognize the 1 cm in one of those dimensions. The last two boxes behave a bit unexpected. Here the depth is added to the height we specified. These last three situations learn us that specifying the height of a `\vtop` does not always make that much sense.

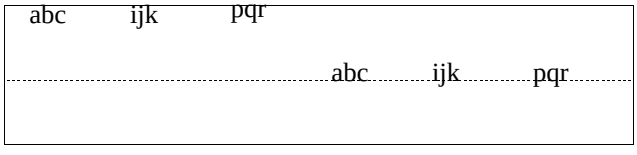
Now watch what happens when we add a `\vss`. This time the ijk and pqr boxes behave as expected and we end up with six boxes of 1 cm. Seeing is believing.

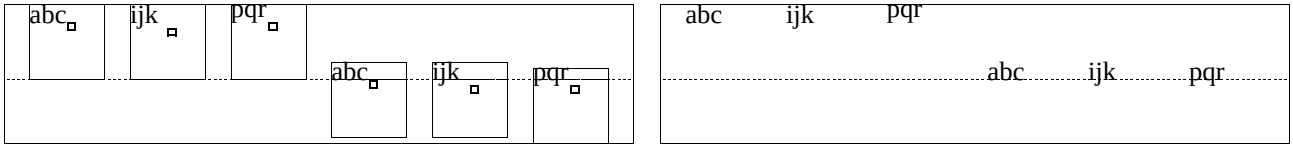




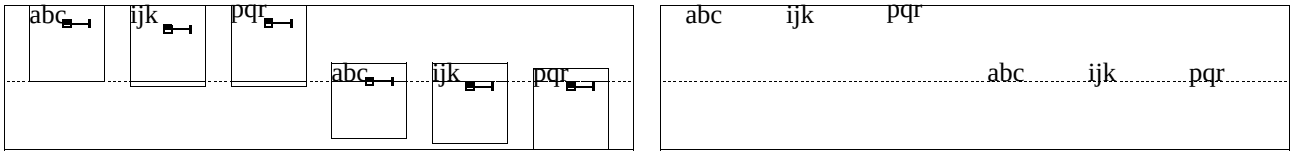
In most cases, one will add some kind of glue to a box, just to get rid of those underfull messages. It's good to be aware of the fact that adding glue does a bit more. Adding a `\vskip` or `\kern` has the same effect.



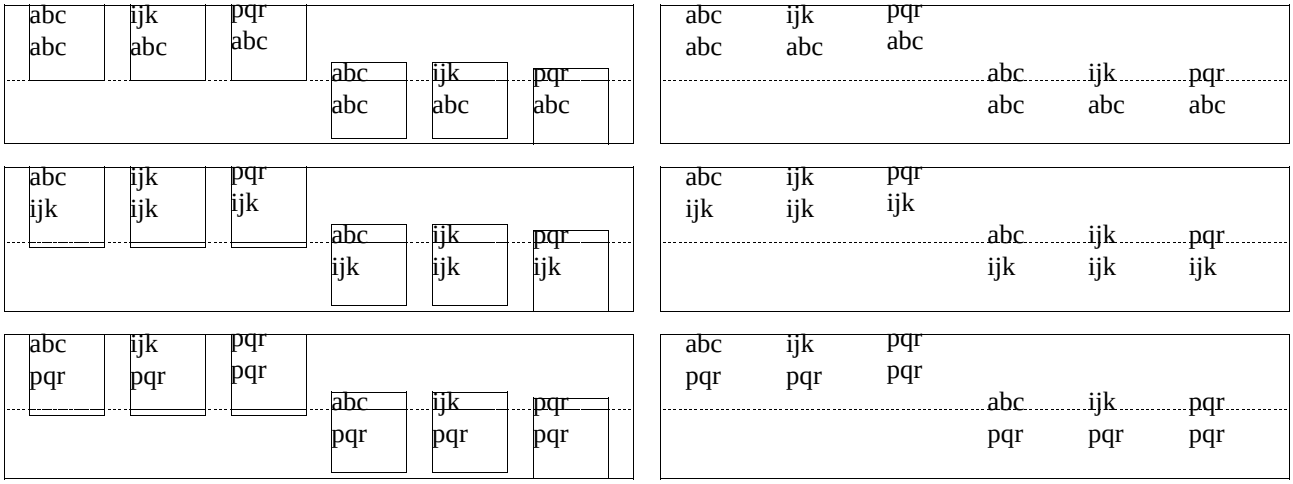




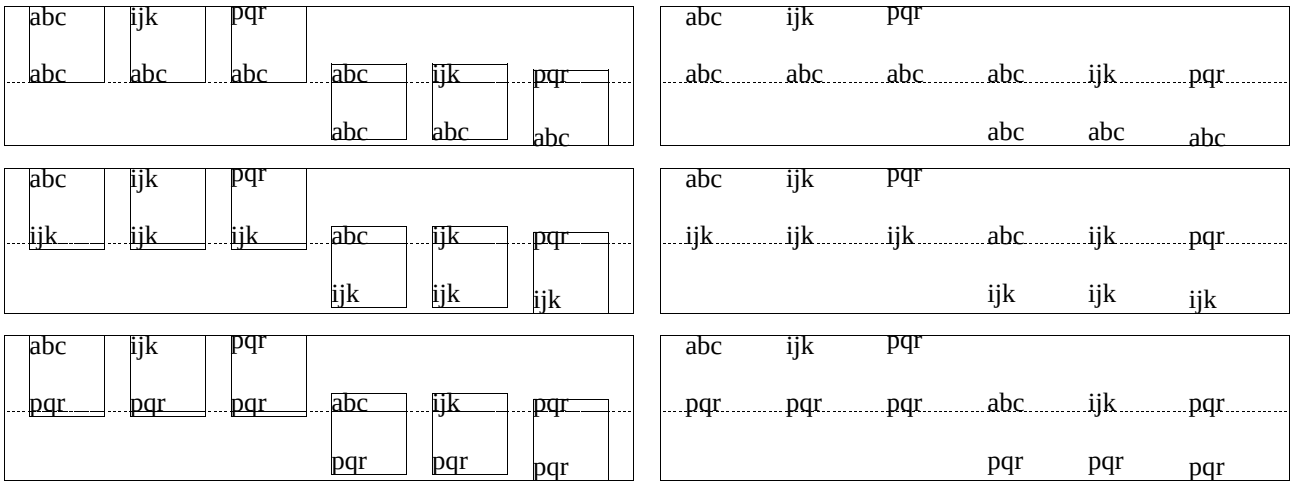
Adding a very large skip or kern makes no difference so we stick to these 3 pt examples. A penalty on the other hand has no effect. Here we get the same results as in the first example.



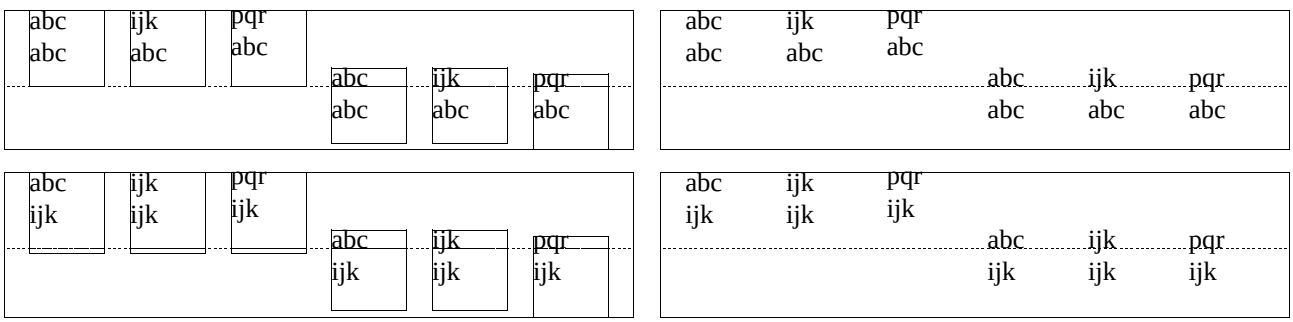
When we add some boxed text, the height and depth of the surrounding box depend on the depth of the (last) line. Here we show what happens when we insert a `\hbox`.

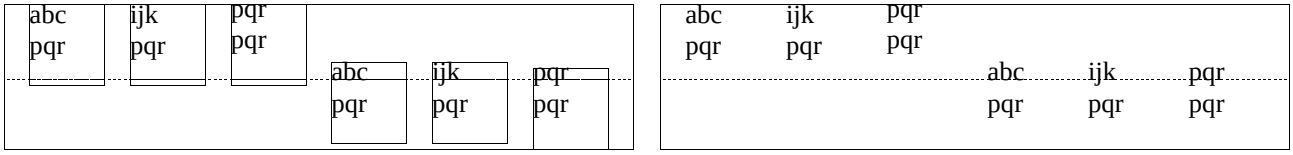


When we put the characters in a `\hbox` and `\unhbox` this box, we get different results. Just take a close look at the next set of boxes.

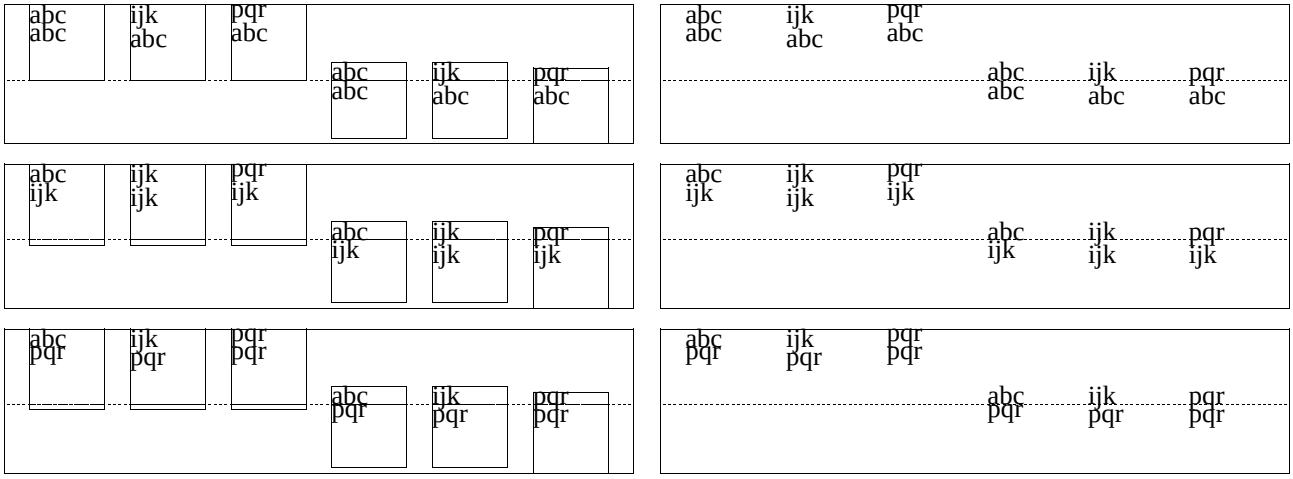


Things looks different when we add a `\vbox`. Just adding one looks like this:





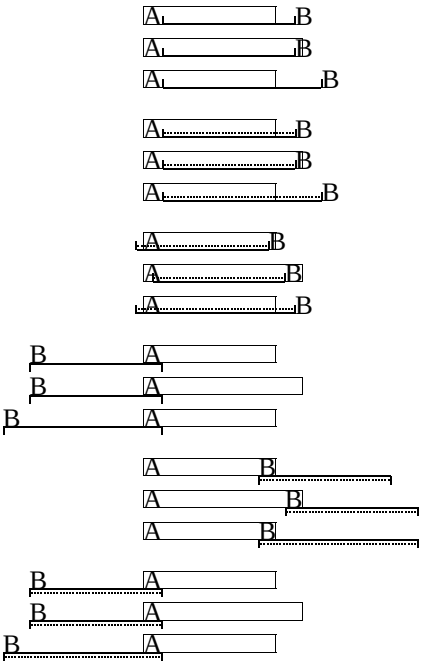
Adding an `\unvbox`'ed one looks a bit different. This kind of tests kan be both very confusing and instructive. It's a chalenge to deduce some systematic behavior from them.



I could show some more examples, like vertical boxes with more lines or `\vtop`'s. The examples shown here at least make clear that when we start manipulating boxes, we have to be aware of side effects.

Close reading of the TEXbook learns that the effects of the skip stretch components `plus` and `minus` sometimes depend on the context. Take a look at the next set of boxes.

```
\hbox to 5em A\hskip 5em B
\hbox to 6em A\hskip 5em B
\hbox to 5em A\hskip 6em B

\hbox to 5em A\hskip 5em plus 1em B
\hbox to 6em A\hskip 5em plus 1em B
\hbox to 5em A\hskip 6em plus 1em B

\hbox to 5em A\hskip 5em minus 1em B
\hbox to 6em A\hskip 5em minus 1em B
\hbox to 5em A\hskip 6em minus 1em B

\hbox to 5em A\hskip -5em B
\hbox to 6em A\hskip -5em B
\hbox to 5em A\hskip -6em B

\hbox to 5em A\hskip -5em plus 1em B
\hbox to 6em A\hskip -5em plus 1em B
\hbox to 5em A\hskip -6em plus 1em B

\hbox to 5em A\hskip -5em minus 1em B
\hbox to 6em A\hskip -5em minus 1em B
\hbox to 5em A\hskip -6em minus 1em B
```



Line- and pagebreaks can in no way be handled 100% perfect. TEX clears out redundant skips and penalties when crossing lines and pages. Making skips and penalties visible calls for the use of boxes and rules. A more perfect visualizer can be build when two more box primitives will be available: `\hnop` and `\vnop`. Both primitives should act like normal boxes when being manipulated, but should be kept out of paragraph and pagebreak calculation. They should be visible in the output but invisible for TEX itself. Lacking these primitives, visualization of sequences of skips and penalties will lead to non-compatible results.

Like the colored verbatim modules described in a previous article, the visual debugger module can be used on top of Plain TEX. Both modules only use a few general system macros, which are supplied in a small miscellaneous module. For CONTEXT users, visualization is always available, because it's just one of the standard features. For users of Plain TEX (or for those who use other packages) the next commands will do the trick:

```
\input supp-vis
```

When this module is loaded, the command `\showmakeup` will turn on the visualization. Users can turn on and off some features, like alignment of vertical cues, individual categories of cues and the visible baseline. The macros and features are explained in detail in the documented module itself.

The `supp` stands for general support. The symmetrical verbatim module, which supports typesetting of colored TEX sources that we presented in a previous article, belongs to this category too. When used outside CONTEXT, both modules automatically fall back on a small module `supp-mis`, which implements poor mans alternatives for a few system macros.

Visualization can best be used grouped. Depending on the number of primitives used, the output can be huge when one processes whole pages. Plain TEX's pagebody routine is both simple and effective. Unfortunately, the more flexibility one wants, the more complicated this routine becomes. In CONTEXT for instance this routine has to deal with multiple headers and footers, backgrounds, logos, multiple margins, interaction menus, navigational tools and a few more. Therefore we turn off visualization as long as we are building the page. The same goes for multi-column handling and some Plain TEX macros like `\llap` and `\rlap`.

In Plain TEX it's not that hard to turn things off temporary. Just give the next code a try:

```
\input supp-vis  \output{\dontshowcomposition\plainoutput}
\showmakeup      \hbox{so much} \eject \hbox{for now} \end
```

In CONTEXT there are some more similar facilities, like general layout, `\strut` and baseline visualization. At the moment, the functionality of this module is limited to the primitives mentioned. We already visualize the mathematical skips, but when needed, we will extend this module with some useful math debugging facilities. A year from now, this module probably will be a bit more advanced anyway.

I could show some more instructive examples, but for producing those, I have to depend a bit too much on CONTEXT for processing. For the same reason the next article, which describes the module itself, lacks some useful functionality.

Let's summarize the cues. Positive horizontal cues are drawn on top of and negative ones under the baseline. The negative cues are drawn in the negative direction. Vertical cues are drawn left or right of the current point (or halfway the `\hsize`) and they too honor the direction. In the next table we only show the horizontal cues.

`\hss`	A B	
`\hfil`	A B	A B
`\hfill`	A B	A B
`\hskip 5em`	A B	B A
`\hskip 5em plus 1em`	A B	B A
`\kern 5em`	A B	B A
`\hglue 5em plus 1em`	A B	B A
`\penalty 200`	A B	A B
`\mskip 50mu plus 1mu`	*A B*	*B A*
`\mkern 50mu`	*A B*	*B A*

Kerns and penalties are treated according to the current mode, which is horizontal or vertical. Zero cues are a special case. A zero horizontal skip for instance shows up as , a kern look like and a zero penalty becomes . As far as possible, different kind of cues add up nicely.

Visual Debugging in TEX

how things are done

Hans Hagen

September 25 1996

Although an integral part of CONTEXT, this module is one of the support modules. Its stand alone character permits use in PLAIN TEX or TEX based macropackages. If in some examples the verbatim listings don't show up nice, this is due to processing by a system that does not support buffering. In CONTEXT we show the commands in the margin, use bit more advanced way of numbering, and typeset the source in TEXnicolored verbatim. Sorry for this inconvenience.

This module is still in development. Depending on my personal need and those of whoever uses it, the macros will be improved in terms of visualization, efficiency and compatibility.

```
1  1  \ifx \undefined \writestatus \input supp-mis.tex \fi
```

One of the strong points of TEX is abstraction of textual input. When macros are defined well and do what we want them to do, we will seldom need the tools present in What You See Is What You Get systems. For instance, when entering text we don't need rulers, because no manual shifting and/or alignment of text is needed. On the other hand, when we are designing macros or specifying layout elements, some insight in TEX's advanced spacing, kerning, filling, boxing and punishment abilities will be handy. That's why we've implemented a mechanism that shows some of the inner secrets of TEX.

```
2  2  \writestatus{loading}{Context Support Macros / Visualization}
```

In this module we are going to redefine some TEX primitives and PLAIN macro's. Their original meaning is saved in macros with corresponding names, preceded by `normal`. These original macros are (1) used to temporary restore the old values when needed and (2) used to prevent recursive calls in the macros that replace them.

```
3  3  \unprotect
```

There are three types of boxes, one horizontal and two vertical in nature. As we will see later on, all three types are to be handled according to their orientation and baseline behavior. Especially `\vtop`'s need our special attention.

```
4  4  \let\normalhbox     = \hbox
   5  \let\normalvbox     = \vbox
   6  \let\normalvtop     = \vtop
```

Next come the flexible skips, which come in two flavors too. Like boxes these are handled with TEX primitives.

```
5  7  \let\normalhskip    = \hskip
   8  \let\normalvskip    = \vskip
```

Both penalties and kerns are taken care of by mode sensitive primitives. This means that when making them visible, we have to take the current mode into account.

```
6  9  \let\normalpenalty  = \penalty
  10  \let\normalkern     = \kern
```

Glues on the other hand are macro's defined in PLAIN TEX. As we will see, their definitions make the implementation of their visible counterparts a bit more TEXnical.

```
7 11  \let\normalhglue    = \hglue
  12  \let\normalvglue    = \vglue
```

Math mode has its own spacing primitives, preceded by `m`. Due to the relation with the current font and the way math is typeset, their unit `mu` is not compatible with other dimensions. As a result, the visual appearance of these primitives is kept primitive too.

```
8 13  \let\normalmkern    = \mkern
  14  \let\normalmskip    = \mskip
```

Fills can be made visible quite easy. We only need some additional negation macros. Because PLAIN TEX only offers `\hfilneg` and `\vfilneg`, we define our own alternative double `ll`'ed ones.

```
\def\hfillneg%
  {\normalhskip\!!zeropoint \!!plus-1fill\relax}

\def\vfillneg%
  {\normalvskip\!!zeropoint \!!plus-1fill\relax}
```

The positive stretch primitives are used independant and in combination with `\leaders`.

```
\let\normalhss      = \hss
\let\normalhfil     = \hfil
\let\normalhfill    = \hfill
\let\normalvss      = \vss
\let\normalvfil     = \vfil
\let\normalvfill    = \vfill
```

Keep in mind that both `\hfillneg` and `\vfillneg` are not part of PLAIN TEX and therefore not documented in standard TEX documentation. They can nevertheless be used at will.

```
\let\normalhfilneg  = \hfilneg
\let\normalhfillneg = \hfillneg
\let\normalvfilneg  = \vfilneg
\let\normalvfillneg = \vfillneg
```

Visualization is not always wanted. Instead of turning this option off in those (unpredictable) situations, we just redefine a few PLAIN macros.

```
\def\rlap#1{\normalhbox to \!!zeropoint{#1\normalhss}}
\def\llap#1{\normalhbox to \!!zeropoint{\normalhss#1}}

\def~{\normalpenalty\!!tenthousand\ }
```

Ruled boxes can be typeset is many ways. Here we present just one alternative. This implementation may be a little complicated, but it supports all three kind of boxes. The next command expects a ⟨*box*⟩ specification, like:

```
\makeruledbox0
```

We can make the baseline of a box visible, both dashed and as a rule. Normally the line is drawn on top of the baseline, but a smashed alternative is offered too. If we want them all, we just say:

```
\baselineruletrue
\baselinefilltrue
\baselinesmashtrue
```

At the cost of some overhead these alternatives are implemented using `\if`'s:

```
\newif\ifbaselinerule  \baselineruletrue
\newif\ifbaselinefill  \baselinefillfalse
\newif\ifbaselinesmash \baselinesmashfalse
```

Rules can be turned on and off, but by default we have:

```
\topruletrue
\bottomruletrue
\leftruletrue
\rightruletrue
```

As we see below:

```
\newif\iftoprule       \topruletrue
\newif\ifbottomrule    \bottomruletrue
\newif\ifleftrule      \leftruletrue
\newif\ifrightrule     \rightruletrue
```

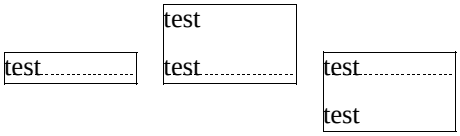
The width in the surrounding rules can be specified by assigning an apropriate value to the dimension used. This module defaults the width to:

```
\boxrulewidth=.2pt
```

Although we are already low on ⟨*dimensions*⟩ it's best to spend one here, mainly because it enables easy manipulation, like multiplication by a given factor.

```
\newdimen\boxrulewidth \boxrulewidth=.2pt
```

The core macro `\makeruledbox` looks a bit hefty. The manipulation at the end is needed because we want to preserve both the mode and the baseline. This means that `\vtop`'s and `\vbox`'es behave the way we expect them to do.



The `\cleaders` part of the macro is responsible for the visual baseline. The `\normalhfill` belongs to this primitive too. By storing and restoring the height and depth of box `#1`, we preserve the mode.

```
\def\makeruledbox#1%
  {\edef\ruledheight {\the\ht#1}%
   \edef\ruleddepth  {\the\dp#1}%
   \edef\ruledwidth  {\the\wd#1}%
   \setbox\scratchbox=\normalvbox
     {\dontcomplain
      \offinterlineskip
      \hrule
        \!!height\boxrulewidth
        \iftoprule\else\!!width\!!zeropoint\fi
      \normalvskip-\boxrulewidth
      \normalhbox to \ruledwidth
         {\vrule
            \!!height\ruledheight
            \!!depth\ruleddepth
            \!!width\ifleftrule\else0\fi\boxrulewidth
          \ifdim\ruledheight>\!!zeropoint \else \baselinerulefalse \fi
          \ifdim\ruleddepth>\!!zeropoint \else \baselinerulefalse \fi
          \ifbaselinerule
            \ifdim\ruledwidth<20\boxrulewidth
              \baselinefilltrue
            \fi
            \cleaders
              \ifbaselinefill
                \hrule
                  \ifbaselinesmash
                    \!!height\boxrulewidth
                  \else
                    \!!height.5\boxrulewidth
                    \!!depth.5\boxrulewidth
                  \fi
              \else
                \normalhbox
                  {\normalhskip2.5\boxrulewidth
                   \vrule
                   \ifbaselinesmash
                     \!!height\boxrulewidth
                   \else
                     \!!height.5\boxrulewidth
                     \!!depth.5\boxrulewidth
                   \fi
                   \!!width5\boxrulewidth
                   \normalhskip2.5\boxrulewidth}%
              \fi
          \fi
          \normalhfill
          \vrule
            \!!width\ifrightrule\else0\fi\boxrulewidth}%
        \normalvskip-\boxrulewidth
        \hrule
          \!!height\boxrulewidth
          \ifbottomrule\else\!!width\!!zeropoint\fi}%
   \wd#1=\!!zeropoint
   \setbox#1=\ifhbox#1\normalhbox\else\normalvbox\fi
     {\normalhbox{\box#1\lower\ruleddepth\box\scratchbox}}%
   \ht#1=\ruledheight
   \wd#1=\ruledwidth
   \dp#1=\ruleddepth}
```

Just in case one didn't notice: the rules are in fact layed over the box. This way the contents of a box cannot visually interfere with the rules around (upon) it. A more advanced version of ruled boxes can be found in one of the core modules of CONTEXT. There we take offsets, color, rounded corners, backgrounds and alignment into account too.

These macro's can be used instead of `\hbox`, `\vbox` and `\vtop`. They just do what their names state. Using an auxiliary macro would save us a few words of memory, but it would make their appearance even more obscure.

one two three four five

```
\hbox {\strut one two \hbox {three} four five}
```

```
19 98  \def\ruledhbox%
   99    {\normalhbox\bgroup
  100     \dowithnextbox{\makeruledbox\nextbox\box\nextbox\egroup}%
  101     \normalhbox}
```

first line
second line
third line
fourth line
fifth line

```
\vbox {\strut first line \par second line \par
third line \par fourth line \par fifth line \strut
}
```

```
20 102 \def\ruledvbox%
  103    {\normalvbox\bgroup
  104     \dowithnextbox{\makeruledbox\nextbox\box\nextbox\egroup}%
  105     \normalvbox}
```

first line
second line
third line
fourth line
fifth line

```
\vtop {\strut first line \par second line \par
third line \par fourth line \par fifth line \strut
}
```

```
21 106 \def\ruledvtop%
  107    {\normalvtop\bgroup
  108     \dowithnextbox{\makeruledbox\nextbox\box\nextbox\egroup}%
  109     \normalvtop}
```

Of the next two macros the first can be used to precede a box of ones own choice. One can for instance prefix boxes with `\ruledbox` and afterwards — when the macro satisfy the needs — let it to `\relax`.

```
\ruledbox\hbox{What rules do you mean?}
```

The macro `\setruledbox` can be used to directly rule a box.

```
\setruledbox12=\hbox{Who's talking about rules here?}
```

At the cost of some extra macros we can implement a variant that does not need the `=`, but we stick to:

```
22 110 \def\ruledbox%
  111    {\dowithnextbox{\makeruledbox\nextbox\box\nextbox}}

23 112 \def\setruledbox#1=%
  113    {\dowithnextbox{\makeruledbox\nextbox\setbox#1=\nextbox}}
```

Before we meet the visualizing macro's, we first implement ourselves some handy utility ones. Just for the sake of efficiency and readability, we introduce some status variables, that tell us a bit more about the registers we use:

```
\ifflexible
\ifzero
\ifnegative
\ifpositive
```

These status variables are set when we call for one of the investigation macros, e.g.

```
\investigateskip\scratchskip
```

We use some dirty trick to check stretchability of ⟨*skips*⟩. Users of these macros are invited to study their exact behavior first. The positive and negative states both include zero and are in fact non-negative (≥ 0) and non-positive (≤ 0) .

```
24 114 \newif\ifflexible
  115 \newif\ifzero
  116 \newif\ifnegative
  117 \newif\ifpositive

25 118 \def\investigateskip#1%
  119    {\relax
  120     \scratchdimen=#1\relax
  121     \edef\!!stringa{\the\scratchdimen}%
  122     \edef\!!stringb{\the#1}%
```

```
    \ifx\!!stringa\!!stringb \flexiblefalse \else \flexibletrue \fi
    \ifdim#1=\!!zeropoint\relax
      \zerotrue      \else
      \zerofalse     \fi
    \ifdim#1<\!!zeropoint\relax
      \positivefalse \else
      \positivetrue  \fi
    \ifdim#1>\!!zeropoint\relax
      \negativefalse \else
      \negativetrue  \fi}
\def\investigatecount#1%
  {\relax
   \flexiblefalse
   \ifnum#1=0
     \zerotrue      \else
     \zerofalse     \fi
   \ifnum#1<0
     \positivefalse \else
     \positivetrue  \fi
   \ifnum#1>0
     \negativefalse \else
     \negativetrue  \fi}
\def\investigatemuskip#1%
  {\relax
   \edef\!!stringa{\the\scratchmuskip}%
   \edef\!!stringb{0mu}%
   \def\!!stringc##1##2\\{##1}%
   \expandafter\edef\expandafter\!!stringc\expandafter
     {\expandafter\!!stringc\!!stringa\\}%
   \edef\!!stringd{-}%
   \flexiblefalse
   \ifx\!!stringa\!!stringb
     \zerotrue
     \negativefalse
     \positivefalse
   \else
     \zerofalse
     \ifx\!!stringc\!!stringd
       \positivefalse
       \negativetrue
     \else
       \positivetrue
       \negativefalse
     \fi
   \fi}
```

Indentation, left and/or right skips, redefinition of `\par` and assignments to `\everypar` can lead to unwanted results. We can therefore turn all those things off with `\dontinterfere`.

```
\def\dontinterfere%
  {\everypar  = {}%
   \let\par   = \endgraf
   \parindent = \!!zeropoint
   \parskip   = \!!zeropoint
   \leftskip  = \!!zeropoint
   \rightskip = \!!zeropoint
   \relax}
```

In this module we do a lot of box manipulations. Because we don't want to be confronted with to many over- and underfull messages we introduce `\dontcomplain`.

```
\def\dontcomplain%
  {\hbadness  = \!!tenthousand
   \hfuzz     = \maxdimen
   \vbadness  = \!!tenthousand
   \vfuzz     = \maxdimen}
```

Now the neccessary utility macros are defined, we can make a start with the visualizing ones. The implementation of these macros is a compromise between readability, efficiency of coding and processing speed. Sometimes we do in steps what could have been done in combination, sometimes we use a few boxes more or less then actually needed, and more than once one can find the same piece of rule drawing code twice.

Depending on the context, one can force visual vertical cues being centered along `\hsize` or being put at the current position. Although centering often looks better, we've chosen the second alternative as default. The main reason for doing so is that often when we don't set the `\hsize` ourselves, TEX takes the value of the surrounding box. As a result the visual cues can migrate outside the current context.

This behavior is accomplished by a small but effective auxiliary macro, which behavior can be influenced by the boolean `\centeredvcue`. By saying

```
\centeredvcuetrue
```

one turns centering on. As said, we turn it off.

```
\newif\ifcenteredvcue  \centeredvcuefalse
\def\normalvcue#1%
  {\normalhbox \ifcenteredvcue to \hsize \fi {\normalhss#1\normalhss}}
```

We could have used the more robust version

```
\def\normalvcue%
  {\normalhbox \ifcenteredvcue to \hsize \fi
   \bgroup\bgroup\normalhss
   \aftergroup\normalhss\aftergroup\egroup
   \let\next=}
```

or the probably best one:

```
\def\normalvcue%
  {\hbox \ifcenteredvcue to \hsize
     \bgroup\bgroup\normalhss
     \aftergroup\normalhss\aftergroup\egroup
   \else
     \bgroup
   \fi
   \let\next=}
```

Because we don't have to preserve ⟨*catcodes*⟩ and only use small arguments, we stick to the first alternative.

We build our visual cues out of rules. At the cost of a much bigger DVI file, this is to be prefered over using characters (1) because we cannot be sure of their availability and (2) because their dimensions are fixed.

As with ruled boxes, we use a ⟨*dimension*⟩ to specify the width of the ruled elements. This dimension defaults to:

```
\testrulewidth=\boxrulewidth
```

Because we prefer whole numbers for specifying the dimensions, we often use even multiples of `\testrulewidth`.

A second variable is introduced because of the stretch components of ⟨*skips*⟩. At the cost of some accuracy we can make this stretch visible.

```
\visiblestretchtrue
```

```
\newdimen\testrulewidth  \testrulewidth=\boxrulewidth
\newif\ifvisiblestretch  \visiblestretchfalse
```

We start with the easiest part, the fills. The scheme we follow is *visual filling – going back – normal filling.* Visualizing is implemented using `\cleaders`. Because the ⟨*box*⟩ that follows this command is constructed only once, the `\copy` is not really a prerequisite. We prefer using a `\normalhbox` here instead of a `\hbox`.

```
\def\setvisiblehfilbox#1\to#2#3#4%
  {\setbox#1=\normalhbox
     {\vrule
        \!!width#2\testrulewidth
        \!!height#3\testrulewidth
        \!!depth#4\testrulewidth}%
   \smashbox#1}
\def\doruledhfiller#1#2#3#4%
  {#1#2%
   \bgroup
     \dontinterfere
     \dontcomplain
     \setvisiblehfilbox0\to{4}{#3}{#4}%
     \setvisiblehfilbox2\to422%
     \copy0\copy2
     \bgroup
```

```
202         \setvisiblehfilbox0\to422%
203         \cleaders
204           \normalhbox to 12\testrulewidth
205             {\normalhss\copy0\normalhss}%
206           #1%
207       \egroup
208       \setbox0=\normalhbox
209         {\normalhskip-4\testrulewidth\copy0\copy2}%
210       \smashbox0
211       \box0
212     \egroup}
```

The horizontal fillers differ in their boundary visualization. Watch the small dots. Fillers can be combined within reasonable margins.

\hss test

\hfil test

\hfill test

\hfil\hfil test \hfil

The negative counterparts are visualizes, but seldom become visible, apart from their boundaries.

\hfilneg test

\hfillneg test

Although leaders are used for visualizing, they are visualized themselves correctly as the next example shows.

All five substitutions use the same auxiliary macro. Watch the positive first – negative next approach.

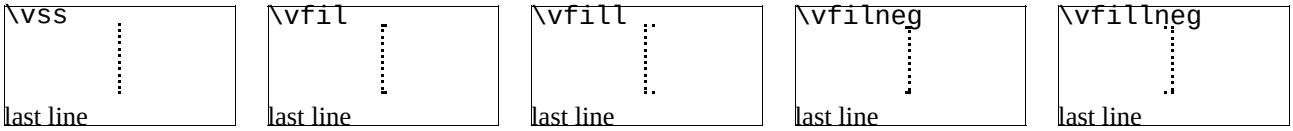
```
35 213 \def\ruledhss%
  214   {\doruledhfiller\normalhss\normalhfilneg{0}{0}}

36 215 \def\ruledhfil%
  216   {\doruledhfiller\normalhfil\normalhfilneg{10}{-6}}

37 217 \def\ruledhfill%
  218   {\doruledhfiller\normalhfill\normalhfillneg{18}{-14}}

38 219 \def\ruledhfilneg%
  220   {\doruledhfiller\normalhfilneg\normalhfil{-6}{10}}

39 221 \def\ruledhfillneg%
  222   {\doruledhfiller\normalhfillneg\normalhfill{-14}{18}}
```

The vertical mode commands adopt the same visualization scheme, but are implemented in a slightly different way.

```
40 223 \def\setvisiblevfilbox#1\to#2#3#4%
  224   {\setbox#1=\normalvcue
  225      {\vrule
  226         \!!width#2\testrulewidth
  227         \!!height#3\testrulewidth
  228         \!!depth#4\testrulewidth}%
  229    \smashbox#1}%

41 230 \def\doruledvfiller#1#2#3%
  231   {#1#2%
  232    \bgroup
  233      \dontinterfere
  234      \dontcomplain
  235      \offinterlineskip
  236      \setvisiblevfilbox0\to422%
  237      \setbox2=\normalhbox
  238        {\normalhskip -#3\testrulewidth\copy0}%
  239      \smashbox2
  240      \copy2
  241      \bgroup
  242        \setbox2=\normalhbox
  243          {\normalhskip -2\testrulewidth\copy0}%
  244        \smashbox2
  245        \copy2
```

```
246         \cleaders
247           \normalvbox to 12\testrulewidth
248             {\normalvss\copy2\normalvss}%
249           #1%
250         \setbox2=\normalvbox
251           {\vskip-2\testrulewidth\copy2}%
252         \smashbox2
253         \box2
254       \egroup
255       \setbox2=\normalvbox
256         {\vskip-2\testrulewidth\copy2}%
257       \smashbox2
258       \box2
259     \egroup}
```

Because they act the same as their horizontal counterparts we only show a few examples.



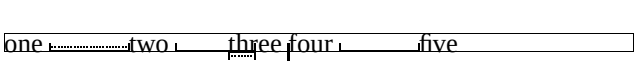
Keep in mind that `\vfillneg` is not part of PLAIN TEX, but are mimmicked by a macro.

```
42 260 \def\ruledvss%
  261   {\doruledvfiller\normalvss\normalvfilneg{2}}
43 262 \def\ruledvfil%
  263   {\doruledvfiller\normalvfil\normalvfilneg{-4}}
44 264 \def\ruledvfill%
  265   {\doruledvfiller\normalvfill\normalvfillneg{-12}}
45 266 \def\ruledvfilneg%
  267   {\doruledvfiller\normalvfilneg\normalvfil{8}}
46 268 \def\ruledvfillneg%
  269   {\doruledvfiller\normalvfillneg\normalvfill{16}}
```

Skips differ from kerns in two important aspects:

- line and pagebreaks are allowed at a skip
- skips can have a positive and/or negative stretchcomponent

Stated a bit different: kerns are fixed skips at which no line or pagebreak can occur. Because skips have a more open character, they are visualized in a open way.




```
one \hskip +30pt plus 5pt two \hskip +30pt \hskip
-10pt plus 5pt three \hskip 0pt four \hskip +30pt
five
```

When skips have a stretch component, this is visualized by means of a dashed line. Positive skips are on top of the baseline, negative ones are below it. This way we can show the combined results. An alternative visualization of stretch could be drawing the mid line over a length of the stretch, in positive or negative direction.

```
47 270 \def\doruledhskip%
  271   {\relax
  272    \dontinterfere
  273    \dontcomplain
  274    \investigateskip\scratchskip
  275    \ifzero
  276      \setbox0=\normalhbox
  277        {\normalhskip-\testrulewidth
  278         \vrule
  279           \!!width4\testrulewidth
  280           \!!height16\testrulewidth
  281           \!!depth16\testrulewidth}%
  282    \else
  283      \setbox0=\normalhbox to \ifnegative-\fi\scratchskip
  284        {\vrule
  285           \!!width2\testrulewidth
  286           \ifnegative\!!depth\else\!!height\fi16\testrulewidth
  287         \cleaders
  288           \hrule
```

```
289             \ifnegative
290               \!!depth2\testrulewidth
291               \!!height\!!zeropoint
292             \else
293               \!!height2\testrulewidth
294               \!!depth\!!zeropoint
295             \fi
296           \normalhfill
297         \ifflexible
298           \normalhskip\ifnegative\else-\fi\scratchskip
299           \normalhskip2\testrulewidth
300           \cleaders
301             \normalhbox
302               {\normalhskip 2\testrulewidth
303                \vrule
304                  \!!width2\testrulewidth
305                  \!!height\ifnegative-7\else9\fi\testrulewidth
306                  \!!depth\ifnegative9\else-7\fi\testrulewidth
307                \normalhskip 2\testrulewidth}%
308             \normalhfill
309         \fi
310         \vrule
311           \!!width2\testrulewidth
312           \ifnegative\!!depth\else\!!height\fi16\testrulewidth}%
313       \setbox0=\normalhbox
314         {\ifnegative\else\normalhskip-\scratchskip\fi
315          \box0}%
316     \fi
317     \smashbox0%
318     \ifvisiblestretch \else
319       \flexiblefalse
320     \fi
321     \ifflexible
322       % breaks ok but small displacements can occur
323       \skip2=\scratchskip
324       \advance\skip2 by -1\scratchskip
325       \divide\skip2 by 2
326       \advance\scratchskip by -\skip2
327       \normalhskip\scratchskip
328       \normalpenalty\!!tenthousand
329       \box0
330       \normalhskip\skip2
331     \else
332       \normalhskip\scratchskip
333       \box0
334     \fi
335     \egroup}
48 336 \def\ruledhskip%
337   {\bgroup
338    \afterassignment\doruledhskip
339    \scratchskip=}
```

The visual skip is located at a feasible point. Normally this does not interfere with the normaltypesetting process. The next examples show (1) the default behavior, (2) the (not entirely correct) distributed stretch and (3) the way the text is typeset without cues.

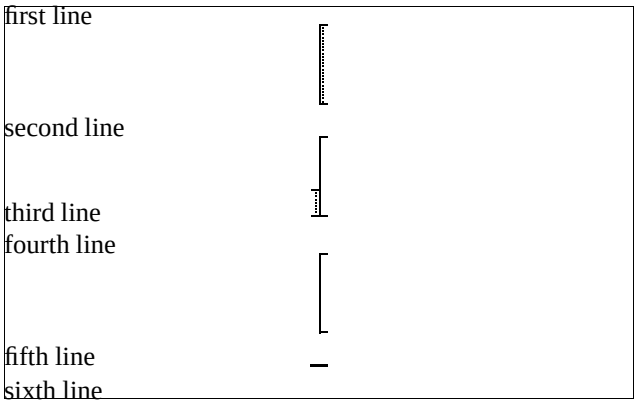
test test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test

test test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test

test test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test testtest test

We are less fortunate when implementing the vertical skips. This is a direct result of interference between the boxes that visualize the skip and skip removal at a pagebreak. Normally skips disappear at the top of a page, but not of course when visualized in a `\vbox`. A quite perfect simulation could have been built if we would have had available two more

primitives: \hnop and \vnop. These new primitives could stand for boxes that are visible but are not taken into account in any way. They are there for us, but not for TEX.




```
first line \vskip +30pt plus 5pt second line
\vskip +30pt \vskip -10pt plus 5pt third line
\par fourth line \vskip +30pt fifth line \vskip
0pt sixth line
```

We have to postpone \prevdepth. Although this precaution probably is not completely waterproof, it works quite well.

```
49 340 \def\dodoruledvskip%
   341   {\nextdepth=\prevdepth
   342    \dontinterfere
   343    \dontcomplain
   344    \offinterlineskip
   345    \investigateskip\scratchskip
   346    \ifzero
   347      \setbox0=\normalvcue
   348        {\vrule
   349           \!!width32\testrulewidth
   350           \!!height2\testrulewidth
   351           \!!depth2\testrulewidth}%
   352    \else
   353      \setbox0=\normalvbox to \ifnegative-\fi\scratchskip
   354        {\hrule
   355           \!!width16\testrulewidth
   356           \!!height2\testrulewidth
   357         \ifflexible
   358           \cleaders
   359             \normalhbox to 16\testrulewidth
   360               {\normalhss
   361                \normalvbox
   362                  {\normalvskip 2\testrulewidth
   363                   \hrule
   364                   \!!width2\testrulewidth
   365                   \!!height2\testrulewidth
   366                   \normalvskip 2\testrulewidth}%
   367                \normalhss}%
   368             \normalvfill
   369         \else
   370           \normalvfill
   371         \fi
   372         \hrule
   373           \!!width16\testrulewidth
   374           \!!height2\testrulewidth}%
   375      \setbox2=\normalvbox to \ht0
   376        {\hrule
   377           \!!width2\testrulewidth
   378           \!!height\ht0}%
   379      \ifnegative
   380        \ht0=\!!zeropoint
   381        \setbox0=\normalhbox
   382          {\normalhskip2\testrulewidth % will be improved
   383           \normalhskip-\wd0\box0}%
   384      \fi
   385      \smashbox0%
   386      \smashbox2%
   387      \setbox0=\normalvcue
   388        {\box2\box0}%
   389      \setbox0=\normalvbox
   390        {\ifnegative\normalvskip\scratchskip\fi\box0}%
```

```
391       \smashbox0%
392     \fi
393     \ifvisiblestretch
394       \ifflexible
395         \skip2=\scratchskip
396         \advance\skip2 by -1\scratchskip
397         \divide\skip2 by 2
398         \advance\scratchskip by -\skip2
399         \normalvskip\skip2
400       \fi
401     \fi
402     \normalpenalty\!!tenthousand
403     \box0
404     \prevdepth=\nextdepth % not \dp0=\nextdepth
405     \normalvskip\scratchskip}
```

We try to avoid interfering at the top of a page. Of course we only do so when we are in the main vertical list.

```
50 406 \def\doruledvskip%
407   {\par
408    \ifdim\pagegoal=\maxdimen
409      \ifinner
410        \dodoruledvskip
411      \fi
412    \else
413      \dodoruledvskip
414    \fi
415    \egroup}
51 416 \def\ruledvskip%
417   {\bgroup
418    \afterassignment\doruledvskip
419    \scratchskip=}
```

The macros that implement the kerns are a bit more complicated than needed, because they also serve the visualization of glue, our PLAIN defined kerns with stretch or shrink. We've implemented both horizontal and vertical kerns as ruled boxes.

one two three four five

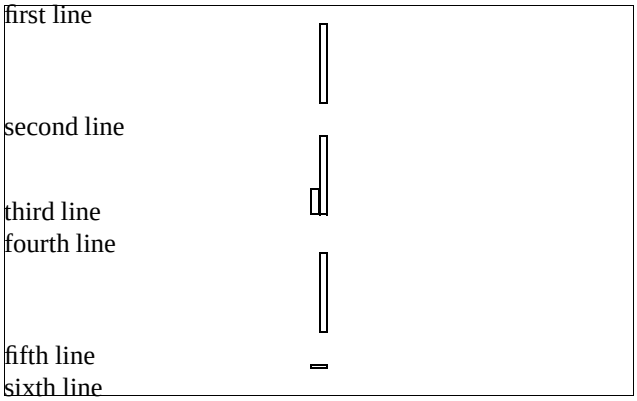
```
one \kern +30pt two \kern +30pt \kern -10pt three
\kern 0pt four \kern +30pt five
```

Positive and negative kerns are placed on top or below the baseline, so we are able to track their added result. We didn't mention spacings of 0 pt yet. Zero values are visualized a bit different, because we want to see them anyhow.

```
52 420 \def\doruledhkern%
421   {\dontinterfere
422    \dontcomplain
423    \baselinerulefalse
424    \investigateskip\scratchskip
425    \boxrulewidth=2\testrulewidth
426    \ifzero
427      \setbox0=\ruledhbox to 8\testrulewidth
428        {\vrule
429           \!!width\!!zeropoint
430           \!!height16\testrulewidth
431           \!!depth16\testrulewidth}%
432      \setbox0=\normalhbox
433        {\normalhskip-4\testrulewidth\box0}%
434    \else
435      \setbox0=\ruledhbox to \ifnegative-\fi\scratchskip
436        {\vrule
437           \!!width\!!zeropoint
438           \ifnegative\!!depth\else\!!height\fi16\testrulewidth
439         \ifflexible
440           \normalhskip2\testrulewidth
441           \cleaders
442             \normalhbox
443               {\normalhskip 2\testrulewidth
444                \vrule
445                  \!!width2\testrulewidth
446                  \!!height\ifnegative-7\else9\fi\testrulewidth
447                  \!!depth\ifnegative9\else-7\fi\testrulewidth
448                \normalhskip 2\testrulewidth}%
```

```
            \normalhfill
          \else
            \normalhfill
          \fi}%
      \testrulewidth=2\testrulewidth
      \setbox0=\ruledhbox{\box0}%  \make...
    \fi
    \smashbox0%
    \normalpenalty\!!tenthousand
    \normalhbox to \!!zeropoint
      {\ifnegative\normalhskip1\scratchskip\fi
       \box0}%
    \afterwards\scratchskip
    \egroup}

\def\ruledhkern#1%
  {\bgroup
   \let\afterwards=#1\relax
   \afterassignment\doruledhkern
   \scratchskip=}
```

After having seen the horizontal ones, the vertical kerns will not surprise us. In this example we use `\par` to switch to vertical mode.




```
first line \par \kern +30pt second line \par
\kern +30pt \par \kern -10pt third line \par
fourth line \par \kern +30pt fifth line \par
\kern 0pt sixth line
```

Like before, we have to postpone `\prevdepth`. If we leave out this trick, we got ourselves some wrong spacing.

```
\def\dodoruledvkern%
  {\nextdepth=\prevdepth
   \dontinterfere
   \dontcomplain
   \baselinerulefalse
   \offinterlineskip
   \investigateskip\scratchskip
   \boxrulewidth=2\testrulewidth
   \ifzero
     \setbox0=\ruledhbox to 32\testrulewidth
       {\vrule
          \!!width\!!zeropoint
          \!!height4\testrulewidth
          \!!depth4\testrulewidth}%
   \else
     \setbox0=\ruledvbox to \ifnegative-\fi\scratchskip
       {\hsize16\testrulewidth
        \ifflexible
          \cleaders
            \normalhbox to 16\testrulewidth
              {\normalhss
               \normalvbox
                 {\normalvskip 2\testrulewidth
                  \hrule
                    \!!width2\testrulewidth
                    \!!height2\testrulewidth
                  \normalvskip 2\testrulewidth}%
               \normalhss}%
            \normalvfill
        \else
          \vrule
```

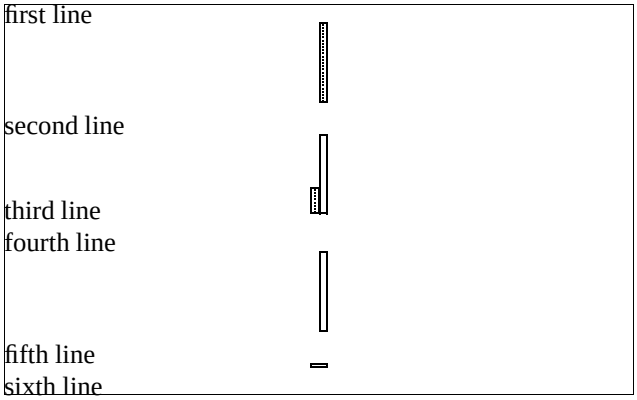
```
            \!!width\!!zeropoint
            \!!height\ifnegative-\fi\scratchskip
          \normalhfill
        \fi}
  \fi
  \testrulewidth=2\testrulewidth
  \setbox0=\ruledvbox{\box0}%  \make...
  \smashbox0%
  \setbox0=\normalvbox
    {\ifnegative\normalvskip\scratchskip\fi
     \normalvcue
      {\ifnegative\normalhskip-16\testrulewidth\fi\box0}}%
  \smashbox0%
  \normalpenalty\!!tenthousand
  \box0
  \prevdepth=\nextdepth} % not \dp0=\nextdepth


\def\doruledvkern%
  {\ifdim\pagegoal=\maxdimen
     \ifinner
       \dodoruledvkern
     \fi
  \else
     \dodoruledvkern
  \fi
  \afterwards\scratchskip
  \egroup}


\def\ruledvkern#1%
  {\bgroup
   \let\afterwards=#1\relax
   \afterassignment\doruledvkern
   \scratchskip=}


\def\ruledkern%
  {\ifvmode
     \let\next=\ruledvkern
  \else
     \let\next=\ruledhkern
  \fi
  \next\normalkern}
```

The non-primitive glue commands are treated as kerns with stretch. This stretch is presented as a dashed line. I have to admit that until now, I've never used these glue commands.

one two three four five

```
one \hglue +30pt plus 5pt two \hglue +30pt \hglue
-10pt plus 5pt three \hglue 0pt four \hglue +30pt
five
```

```
\def\doruledhglue%
  {\leavevmode
   \scratchcounter=\spacefactor
   \vrule\!!width\!!zeropoint
   \normalpenalty\!!tenthousand
   \ruledhkern\normalhskip\scratchskip
   \spacefactor=\scratchcounter
   \egroup}


\def\ruledhglue%
  {\bgroup
   \afterassignment\doruledhglue\scratchskip=}
```




```
first line \vglue +30pt plus 5pt second line
\vglue +30pt \vglue -10pt plus 5pt third line
\par fourth line \vglue +30pt fifth line \vglue
0pt sixth line
```

```
\def\doruledvglue%
  {\par
   \nextdepth=\prevdepth
   \hrule\!!height\!!zeropoint
   \normalpenalty\!!tenthousand
   \ruledvkern\normalvskip\scratchskip
   \prevdepth=\nextdepth
   \egroup}
\def\ruledvglue%
  {\bgroup
   \afterassignment\doruledvglue\scratchskip=}
```

Mathematical kerns and skips are specified in mu. This font related unit is incompatible with those of ⟨*dimensions*⟩ and ⟨*skips*⟩. Because in math mode spacing is often a very subtle matter, we've used a very simple, not overloaded way to show them.

```
\def\dodoruledmkern#1%
  {\dontinterfere
   \dontcomplain
   \setbox0=\normalhbox
     {$\normalmkern\ifnegative-\fi\scratchmuskip$}%
   \setbox0=\normalhbox to \wd0
     {\vrule
        \!!height16\testrulewidth
        \!!depth16\testrulewidth
        \!!width\testrulewidth
      \leaders
        \hrule
          \!!height\ifpositive16\else-14\fi\testrulewidth
          \!!depth\ifpositive-14\else16\fi\testrulewidth
        \normalhfill
      \ifflexible
        \normalhskip-\wd0
        \leaders
          \hrule
            \!!height\testrulewidth
            \!!depth\testrulewidth
          \normalhfill
      \fi
      \vrule
        \!!height16\testrulewidth
        \!!depth16\testrulewidth
        \!!width\testrulewidth}%
   \smashbox0%
   \ifnegative
      #1\scratchmuskip
     \box0
   \else
     \box0
      #1\scratchmuskip
   \fi
   \egroup}
```

$a = b + c$

```
$a \mkern 3mu = \mkern 3mu b \quad \mkern -2mu
+ \mkern -2mu \quad c$
```

```
\def\doruledmkern%
```

```
  {\investigatemuskip\scratchmuskip
   \flexiblefalse
   \dodoruledmkern\normalmkern}

\def\ruledmkern%
  {\bgroup
   \afterassignment\doruledmkern\scratchmuskip=}
```

$a \mskip 3mu = \mskip 3mu b \quad \mskip -2mu + \mskip -2mu \quad c$

```
\def\doruledmskip%
  {\investigatemuskip\scratchmuskip
   \flexibletrue
   \dodoruledmkern\normalmskip}

\def\ruledmskip%
  {\bgroup
   \afterassignment\doruledmskip\scratchmuskip=}
```

After presenting fills, skip, kerns and glue we've come to see penalties. In the first implementation — most of the time needed to develop this set of macros went into testing different types of visualization — penalties were mere small blocks with one black half, depending on the sign. This most recent version also gives an indication of the amount of penalty. Penalties can go from less than −10000 to over +10000, and their behavior is somewhat non-lineair, with some values having special meanings. We therefore decided not to use its value for a lineair indicator.

one two three four five

```
one \penalty +100 two \penalty +100 \penalty
-100 three \penalty 0 four \penalty +100 five
```

The small sticks at the side of the penalty indicate it size. The next example shows the positive and negative penalties of 0, 1, 10, 100, 1000 and 10000.

test test test test test test test

test test test test test test test

This way stacked penalties of different severance can be shown in combination.

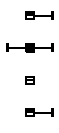
test test test test

```
\def\setruledpenaltybox#1#2#3#4#5#6%
  {\setbox#1=\normalhbox
     {\ifnum#2=0 \else
        \ifnum#2>0
          \def\sign{+}%
        \else
          \def\sign{-}%
        \fi
        \dimen0=\ifnum\sign#2>9999
                  28\else
                \ifnum\sign#2>999
                  22\else
                \ifnum\sign#2>99
                  16\else
                \ifnum\sign#2>9
                  10\else
                  4
                \fi\fi\fi\fi \testrulewidth
        \ifnum#2<0
          \normalhskip-\dimen0
          \normalhskip-2\testrulewidth
          \vrule
            \!!width2\testrulewidth
            \!!height#3\testrulewidth
            \!!depth#4\testrulewidth
        \fi
        \vrule
          \!!width\dimen0
          \!!height#5\testrulewidth
          \!!depth#6\testrulewidth
        \ifnum#2>0
          \vrule
            \!!width2\testrulewidth
            \!!height#3\testrulewidth
```

```
643             \!!depth#4\testrulewidth
644           \fi
645       \fi}%
646     \smashbox#1}
68 647 \def\doruledhpenalty%
648   {\dontinterfere
649    \dontcomplain
650    \investigatecount\scratchcounter
651    \testrulewidth=2\testrulewidth
652    \boxrulewidth=\testrulewidth
653    \setbox0=\ruledhbox to 8\testrulewidth
654      {\ifnegative\else\normalhss\fi
655       \vrule
656         \!!depth8\testrulewidth
657         \!!width\ifzero0\else4\fi\testrulewidth
658       \ifpositive\else\normalhss\fi}%
659    \setruledpenaltybox{2}{\scratchcounter}{0}{8}{-3.5}{4.5}%
660    \normalpenalty\!!tenthousand
661    \setbox0=\normalhbox
662      {\normalhskip-4\testrulewidth
663       \ifnegative
664         \box2\box0
665       \else
666         \box0\box2
667       \fi}%
668    \smashbox0%
669    \box0
670    \normalpenalty\scratchcounter
671    \egroup}
69 672 \def\ruledhpenalty%
673   {\bgroup
674    \afterassignment\doruledhpenalty
675    \scratchcounter=}
```

The size of a vertical penalty is also shown on the horizontal axis. This way there is less interference with the often preceding or following skips and kerns.

first line
second line
third line
fourth line
fifth line



```
first line \par \penalty +100 second line \par
\penalty +100 \par \penalty -100 third line \par
\penalty 0 fourth line \par \penalty +100 fifth
line
```

```
70 676 \def\doruledvpenalty%
677   {\ifdim\pagegoal=\maxdimen
678    \else
679      \nextdepth=\prevdepth
680      \dontinterfere
681      \dontcomplain
682      \investigatecount\scratchcounter
683      \testrulewidth=2\testrulewidth
684      \boxrulewidth=\testrulewidth
685      \setbox0=\ruledhbox
686        {\vrule
687           \!!height4\testrulewidth
688           \!!depth4\testrulewidth
689           \!!width\!!zeropoint
690         \vrule
691           \!!height\ifnegative.5\else4\fi\testrulewidth
692           \!!depth\ifpositive.5\else4\fi\testrulewidth
693           \!!width8\testrulewidth}%
694      \setruledpenaltybox{2}{\scratchcounter}{4}{4}{.5}{.5}%
695      \setbox0=\normalhbox
696        {\normalhskip-4\testrulewidth
697         \ifnegative
698           \box2\box0
699         \else
700           \box0\box2
701         \fi
702         \normalhss}%
```

```
      \smashbox0%
      \normalpenalty\!!tenthousand
      \nointerlineskip
      \dp0=\nextdepth  % not \prevdepth=\nextdepth
      \normalvbox
        {\normalvcue{\box0}}%
    \fi
    \normalpenalty\scratchcounter
    \egroup}

\def\ruledvpenalty%
  {\bgroup
   \afterassignment\doruledvpenalty
   \scratchcounter=}

\def\ruledpenalty%
  {\ifvmode
     \let\next=\ruledvpenalty
   \else
     \let\next=\ruledhpenalty
   \fi
   \next}
```

For those who want to manipulate the visual cues in detail, we have grouped them.

```
\def\showfils%
  {\let\hss      = \ruledhss
   \let\hfil     = \ruledhfil
   \let\hfill    = \ruledhfill
   \let\hfilneg  = \ruledhfilneg
   \let\hfillneg = \ruledhfillneg
   \let\vss      = \ruledvss
   \let\vfil     = \ruledvfil
   \let\vfill    = \ruledvfill
   \let\vfilneg  = \ruledvfilneg
   \let\vfillneg = \ruledvfillneg}

\def\dontshowfils%
  {\let\hss      = \normalhss
   \let\hfil     = \normalhfil
   \let\hfill    = \normalhfill
   \let\hfilneg  = \normalhfilneg
   \let\hfillneg = \normalhfillneg
   \let\vss      = \normalvss
   \let\vfil     = \normalvfil
   \let\vfill    = \normalvfill
   \let\vfilneg  = \normalvfilneg
   \let\vfillneg = \normalvfillneg}

\def\showboxes%
  {\baselineruletrue
   \let\hbox     = \ruledhbox
   \let\vbox     = \ruledvbox
   \let\vtop     = \ruledvtop}

\def\dontshowboxes%
  {\let\hbox     = \normalhbox
   \let\vbox     = \normalvbox
   \let\vtop     = \normalvtop}

\def\showskips%
  {\let\hskip    = \ruledhskip
   \let\vskip    = \ruledvskip
   \let\kern     = \ruledkern
   \let\mskip    = \ruledmskip
   \let\mkern    = \ruledmkern
   \let\hglue    = \ruledhglue
   \let\vglue    = \ruledvglue}

\def\dontshowskips%
  {\let\hskip    = \normalhskip
   \let\vskip    = \normalvskip
   \let\kern     = \normalkern
   \let\mskip    = \normalmskip
```

```
    \let\mkern    = \normalmkern
    \let\hglue    = \normalhglue
    \let\vglue    = \normalvglue}

\def\showpenalties%
  {\let\penalty  = \ruledpenalty}

\def\dontshowpenalties%
  {\let\penalty  = \normalpenalty}
```

All these nice options come together in two macros. The first one turns the options on, the second turnes them off. Both macros only do their job when we are actually showing the composition.

```
\showingcompositiontrue
\showcomposition
```

Because the output routine can do tricky things, like multiple column typesetting and manipulation of the pagebody, shifting things around and so on, the macro `\dontshowcomposition` best can be called when we enter this routine. Too much visual cues just don't make sense. In CONTEXT this has been taken care of.

```
\newif\ifshowingcomposition

\def\showcomposition%
  {\ifshowingcomposition
     \showfils
     \showboxes
     \showskips
     \showpenalties
   \fi}

\def\dontshowcomposition%
  {\ifshowingcomposition
     \dontshowfils
     \dontshowboxes
     \dontshowskips
     \dontshowpenalties
   \fi}
```

Just to make things even more easy, we have defined:

```
\showmakeup
```

For the sake of those who don't (yet) use CONTEXT we preset `\defaulttestrulewidth` to the already set value. Otherwise we default to a corps related value.

```
\def\defaulttestrulewidth{.2pt}
```

Beware, it's a macro not a ⟨*dimension*⟩.

```
\ifx\korpsgrootte\undefined
  \edef\defaulttestrulewidth{\the\testrulewidth}
\else
  \def\defaulttestrulewidth{.02\korpsgrootte}  % still dutch
\fi

\def\showmakeup%
  {\testrulewidth=\defaulttestrulewidth
   \showingcompositiontrue
   \showcomposition}

\protect
```

The documented source you have been reading was processed using some surrogate makeup. When this file is processed in CONTEXT, a few more examples show up here, like a local table of contents and a local register.

Where to find CONTEXT

Hans Hagen

September 25 1996

This year we start uploading (parts of) CONTEXT, a full featured, parameter driven, integrated and stable macro package, to the CTAN distribution sites. We will at least support the dutch and english user interface (commands, parameters, messages), but when needed, we can define some more.

As an independant package CONTEXT has got its own entry at CTAN. The `context` directory will be structured as:

```
/context/ppchtex
/context/generic
/context/sources
/context/extras
/context/documents
```

At the moment, we've uploaded the stand alone chemical module PPCHTEX, and when you read this, the verbatim and debugger modules will be located under `generic`. The full CONTEXT distribution will be located under `sources` and additional modules under `extras`.

Documentation, i.e. the 450+ pages paper users manual as well as the interactive manual, which is cross linked to its source, will be under `documents`. At this moment we're half way documenting the source. The complete source itself will be available as soon as the documentation is finished. We can be reached by e-mail at `pragma@pi.net`.