MAPS

NUMMER 28 • NAJAAR 2002

NTG NEDERLANDSTALIGE TEX GEBRUIKERSGROEP

Voorzitter
Hans Hagen
pragma@wxs.nl

Secretaris
Jules E van Weerden
Jules.vanWeerden@vet.uu.nl

Penningmeester
Wybo Dekker
Servalys, Deil
wybo@servalys.nl

Bestuursleden
Erik Frambach
erik.frambach@stelvio.nl

Frans Goddijn
frans@iaf.nl

Karel H Wesseling
k.h.wesseling@planet.nl

Postadres
Nederlandstalige TeX Gebruikersgroep
Utrechtsestraatweg 64
3445 AV Woerden

Postgiro
1306238
t.n.v. NTG, Deil
000-1662209-17
t.n.v. NTG, Koekelare, België

E-mail bestuur
ntg@ntg.nl

E-mail MAPS redaktie
maps@ntg.nl

WWW
www.ntg.nl

De **Nederlandstalige TeX Gebruikersgroep (NTG)** is een vereniging die tot doel heeft het bevorderen van de kennis en het gebruik van TeX. De NTG fungeert als een forum voor nieuwe ontwikkelingen met betrekking tot computergebaseerde document-opmaak in het algemeen en de ontwikkeling van 'TeX and friends' in het bijzonder. De doelstellingen probeert de NTG te realiseren door onder meer het uitwisselen van informatie, het organiseren van conferenties en symposia m.b.t. TeX en 'TeX-producten'.
De NTG biedt haar leden onder andere het volgende:

- Tweemaal per jaar een NTG-bijeenkomst.
- Het NTG-tijdschrift MAPS.
- De 'TeX Live CD-ROM' en de 'CTAN CD-ROM' met de complete TeX software-archieven.
- Verschillende discussielijsten (mailing lists) over TeX-gerelateerde onderwerpen, zowel voor beginners als gevorderden, algemeen en specialistisch.
- De FTP server ftp.ntg.nl waarop vele honderden megabytes aan algemeen te gebruiken 'TeX-producten' staan.
- De WWW server www.ntg.nl waarop algemene informatie staat over de NTG, bijeenkomsten, publicaties, en links naar andere TeX sites.
- Korting op (buitenlandse) TeX-conferenties en -cursussen en op het lidmaatschap van andere TeX-gebruikersgroepen.

Lid worden kan door overmaking van de verschuldigde contributie naar de NTG-giro (zie links); daarnaast dient via www.ntg.nl een informatieformulier te worden ingevuld. Zonodig kan ook een papieren formulier bij het secretariaat worden opgevraagd.
De contributie bedraagt € 40; voor studenten geldt een tarief van € 20. Dit geeft alle lidmaatschapsvoordelen maar *geen stemrecht*. Een bewijs van inschrijving is vereist. Een gecombineerd NTG/TUG-lidmaatschap levert een korting van 10% op beide contributies op. De prijs in euro's wordt bepaald door de dollarkoers aan het begin van het jaar. De ongekorte TUG-contributie is momenteel $65.

MAPS bijdragen kunt u opsturen naar maps@ntg.nl, bij voorkeur in LaTeX- of Context formaat. Bijdragen op alle niveaus van expertise zijn welkom.

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 producten, waarmee ook de logische structuur van een document beschreven kan worden, met behoud van de letterzet-mogelijkheden van TeX. Voorbeelden zijn LaTeX van Leslie Lamport, $\mathcal{AMS}$-TeX van Michael Spivak, en ConTeXt van Hans Hagen.

Inhoudsopgave

Nummer 28, najaar 2002

Redactioneel

Johannes Braams
`johannes@braams.cistron.nl`

Voor je ligt het achtentwingste nummer van de Maps. Ik wil graag beginnen met mijn excuses uit te spreken voor het vorige nummer. Daaraan zijn een aantal zaken misgegaan. Allereerst wil ik me verontschuldigen bij de auteurs wier artikelen niet of onvolledig gepubliceerd zijn.

- In het artikel van Hans Hagen over MathML ontbraken 4 voorbeeld listings. We hebben ze in dit nummer als erratum opgenomen.
- Patrick Gundlach had van ons een letter extra in zijn achternaam gekregen. Bovendien misten in zijn artikel over het euro symbool twee belangrijke afbeeldingen. Daarom hebben we gemeend het artikel integraal opnieuw op te moeten nemen.
- Een artikel van Fabrice Popineau, dat door Karel H Wesseling samen met zijn vrouw uit het frans is vertaald, was abusievelijk niet afgedrukt. Dat artikel, „Practical MetaPost" is in dit nummer alsnog opgenomen.
- De handleiding van „texexec", die door Hans Hagen en Karel H Wesseling was geschreven heeft de gedrukte versie ook niet gehaald. Ook dit artikel wordt in dit nummer alsnog gepubliceerd.

Daarnaast vind ik een verontschuldiging aan de lezers ook wel op zijn plaats die wellicht gezocht hebben naar afbeeldingen en die niet hebben kunnen vinden. Ik hoop dat we dergelijke blunders vanaf nu niet meer zullen maken onder het motto „van je fouten moet je leren".

Alle fouten in het vorige nummer hebben Karel H Wesseling ertoe doen besluiten zijn redacteurschap met onmiddelijke ingang neer te leggen.

Tot onze schrik is onze mederedacteur Piet van Oostrum in september in het ziekenhuis opgenomen voor een zware operatie. Ik wil hem vanaf deze plaats veel sterkte toewensen en hoop dat hij geheel zal kunnen herstellen.

Het gevolg van dit alles is wél dat de redactie bij het samenstellen en produceren van dit nummer enigszins onderbemand is geweest. Wij willen daarom mensen, die denken dat ze een enthousiaste bijdrage aan het werk van de redactie willen en kunnen leveren, van harte uitnodigen zich bij ons (of bij het bestuur) te melden opdat we de voorjaarseditie van 2003 weer met een redactie op volle sterkte kunnen voorbereiden.

Tenslotte wil ik nog even vooruitblikken naar de verdere inhoud van deze MAPS.

Siep heeft een nieuwe „flyer" gemaakt ter vervanging van de oude. Je treft een afdruk van de flyer aan in de MAPS samen met een korte toelichting.

Michael Guravage is deze zomer afgereisd naar India en heeft daar de TUG conferentie bijgewoond en heeft voor ons een verslag geschreven. Wie meer wil weten over die conferentie kan terecht op `http://www.tug.org.in/tug2002`. In deze MAPS zul je ook de call for papers aantreffen voor de TUG conferentie van 2003. Dat belooft een bijzondere te worden, TeX viert dan zijn 25e verjaardag op Hawaii.

Vóórdat Piet in het ziekenhuis werd opgenomen heeft hij nog wel even voor ons een blik in het CTAN archief geworpen en een aantal recente vernieuwingen beschreven.

Naast het al genoemde artikel over het werken met MetaPost vindt je nog twee artikelen over dit onderwerp, één van Frans Goddijn en Karel H Wesseling over het maken van grafieken, en één van Karel H Wesseling over zijn briefpapier logo.

Wybo Dekker deelt met ons zijn kennis op het gebied van scripts. Hij heeft een soort „make" (`mkl`) voor LaTeX documenten in perl ontwikkeld én hij heeft een hulp bij het bekijken van documenten (`vpp`) ontwikkeld. Tevens presenteert Wybo een oplossing voor gecentreerde, drijvende, objecten mét voetnoten.

Maarten Wisse is bezig geweest met TeX4ht en XML en vertelt over zijn ervaringen en resultaten op dat gebied.

Colofon

De Maps wordt gezet met behulp van een LaTeX class file en een ConTeXt module. De gebruikte fonts zijn Adobe Times-Roman met Old Style Figures en Frutiger, met een versmalde Courier voor de 'verbatim' omgevingen. Compilatie gebeurt met web2c versie 7.3.3.7 onder Linux 2.4 en we drukken op 90-grams gecoat papier vanaf een PDF bestand dat wordt gemaakt met pdf[e]tex 1.00b en dvips versie 5.90a, gedestilleerd met Adobe's Distiller 4.05, draaiend onder Windows NT.

Siep Kroonenberg
siepo@cybercomm.nl

ntg nieuws

De NTG flyer

De flyer die u in deze MAPS aantreft is bedoeld om zieltjes te winnen. Als u denkt dat er mensen zijn in uw omgeving die beter TEX zouden kunnen gebruiken, laat ze dan deze flyer zien, of, nog beter, vraag bij het bestuur (email: ntg@ntg.nl gratis een stapeltje aan, voor op de leestafel van uw afdeling of als bijlage bij uw afdelingskrantje: we hebben er een heleboel laten drukken.

Iets over de produktie:

- De flyer is opgemaakt in LATEX. De voorkant is geïmplementeerd als een blokkendoos van minipages binnen minipages.
- De gebruikte TEX-implementatie was een vanaf source gecompileerde teTeX draaiend onder Debian Linux.
- Voor het compileren van de voorbeelden werd gebruik gemaakt van pdfscreen, ArabTEX, xypic en MetaPost.
- Het Lilypond-voorbeeld is niet door mijzelf gecompileerd maar is uit de PostScript-versie van de handleiding gelicht. Dit voorbeeld is, in verband met het voorkomen van bitmapped fonts, omgezet naar een bitmap en in de GIMP op maat gesneden.
- Conversies tussen PostScript en pdf heb ik uitgevoerd met pdftops uit de xpdf distributie, met Ghostscript en met de op Ghostscript gebaseerde script epstopdf uit de teTEX-distributie.
- Het pdfscreen voorbeeld is omgezet van rgb-kleur naar zwart en steunkleur met gebruikmaking van dvips en een aangepaste versie van colorsep.pro, Sebastian Rahtz' PostScript headerfile voor kleurseparatie.

Er is ook commerciële software gebruikt:

- De tekst- en titelfonts zijn respektievelijk Frutiger Light en Frutiger Bold van Adobe.
- Het NTG-logo is lang geleden gemaakt met Adobe Illustrator versie 7 voor Windows.
- De uiteindelijke pdf is gegenereerd met Adobe Distiller versie 4.05 draaiend onder Windows NT draaiend onder VMware 3 draaiend onder Linux. De drukker zorgde voor kleurscheiding.

NEDERLANDSTALIGE TEX GEBRUIKERSGROEP

TEX: meer dan tekstverwerken

TEX is een elektronisch zetsysteem. Oorspronkelijk speciaal voor wiskundig zetwerk ontwikkeld door Prof. Donald E. Knuth, wordt TEX nu gebruikt voor vrijwel alle denkbare tekstuele toepassingen: brieven, boeken, handleidingen, database uitvoer, flyers zoals deze, presentaties en interactieve documenten. TEX is gratis en is beschikbaar voor vrijwel alle computer-platforms. Er is een levendige online gebruikersgemeenschap.

TEX werkt met opmaakcodes, net als html/xml. Een voorbeeld van TEX-invoer:

```
\documentclass{article}
\begin{document}
Hello world!\par
$ \phi = \sqrt x $
\end{document}
```

Hello world!
$\phi = \sqrt{x}$

TEX maakt uit deze invoer naar keuze een pdf-bestand of een afdruk op papier.

Met een wysiwyg tekstverwerker bent u waarschijnlijk sneller aan de slag, maar als u mocht besluiten de sprong naar TEX te wagen dan zult u merken dat TEX een praktisch en efficiënt stuk gereedschap is met enorme mogelijkheden.

TEX is betrouwbaar

Technieken die goed werken voor een document van vijf pagina's werken even goed voor een boek van vijfhonderd. De extra inwerktijd verdient u ruimschoots terug omdat TEX gewoon zijn werk blijft doen, hoe lang en complex het dokument ook wordt.

TEX is professioneel

TEX levert kwaliteitszetwerk af, dankzij ondersteuning van kerning en ligaturen, en gebruik van het beste afbreekalgoritme ter wereld. TEX kan goede kwaliteit PostScript en pdf genereren. TEX is dan ook in gebruik bij wetenschappelijke uitgevers.

TEX is aktueel

- Rechtstreekse ondersteuning van pdf, met inbegrip van de interactieve voorzieningen hiervan. Met pdflatex en het pdfscreen-pakket kunt u gemakkelijk presentaties maken. Met ConTEXt is er nog veel meer mogelijk.
- XML/SGML: TEX wordt hier ingezet voor het genereren van pdf- en gedrukte uitvoer. Pakketten zoals xmlTEX en ConTEXt kunnen xml rechtstreeks verwerken.
- Het Omega-project werkt aan ondersteuning voor Unicode.

TEX kan in specialistische behoeften voorzien

Dankzij het open karakter en de programmeerbaarheid van TEX kunnen slimme TEX-gebruikers oplossingen bedenken voor hun specifieke problemen, die ze dan aan de gemeenschap ter beschikking stellen. Hieraan hebben wij het bestaan te danken van BibTEX voor het automatisch genereren van bibliografieën uit een database, en van Makeindex voor het automatisch genereren van registers. Voor niet-westerse talen, bladmuziek, presentaties en diagrammen zijn oplossingen van hoog niveau ontwikkeld voor gebruikers door gebruikers, getuige de voorbeelden op deze pagina.

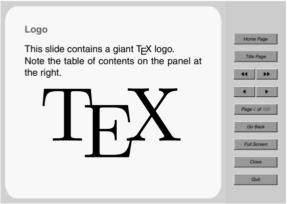
TEX leent zich voor automatische verwerking

In Unix-omgevingen wordt TEX dan ook dankbaar ingezet om automatisch pdf te genereren uit bijvoordeeld DocBook (Linux Documentation Project) of texinfo (het bronformaat voor documentatie voor GNU software).

Herkomst van de figuren. Het arabtex voorbeeld is ontleend aan de ArabTEX documentatie. Het is de aanhef van een traditionele vertelling. Het xypic diagram en het lilypond muziekfragment zijn eveneens ontleend aan de documentatie van de corresponderende pakketten. De MetaPost-figuur is gebaseerd op een bijdrage van Huib van de Stadt.

$$\sum_{i=1}^{qT-1} u_i^2 + \sum_{i=0}^{T-1} \mu_i \left\{ \left[\sum_{j=0}^{q-1} x_j u_{iq+j} \right] - v_i \right\}$$

Wiskunde



Logo

This slide contains a giant TEX logo.
Note the table of contents on the panel at the right.

TEX

pdfscreen

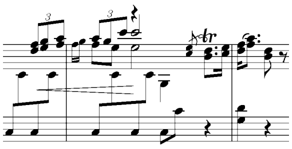
جُحَا وَحَمِيرُهُ العَشَرَةُ

اِشتَرَى جُحَا عَشَرَةَ حَمِيرٍ. فَرِحَ بِهَا وَسَاقَهَا أَمَامَهُ، ثُمَّ رَكِبَ
وَاحِدًا مِنهَا. وَفِي الطَّرِيقِ عَدَّ حَمِيرَهُ وَهُوَ رَاكِبٌ، فَوَجَدَهَا
تِسعَةً. ثُمَّ نَزَلَ وَعَدَّهَا فَرَآهَا عَشَرَةٌ فَقَالَ:
أَمشِي وَأَكسِبُ حِمَارًا، أَفضَلُ مِن أَن أَركَبَ وَأَخسَرَ حِمَارًا.

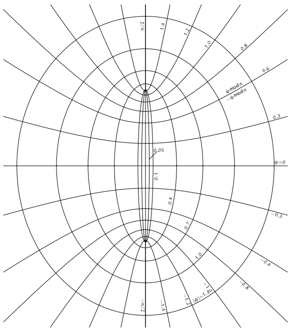
ArabTEX

$$\begin{tikzcd}
U \arrow[drr, bend left, "x"] \arrow[dr, dotted, "(x.y)"] \arrow[ddr, bend right, "y"'] & & \\
& X \times_Z Y \arrow[r, "p"] \arrow[d, "q"] & X \arrow[d, "f"] \\
& Y \arrow[r, "g"] & Z
\end{tikzcd}$$

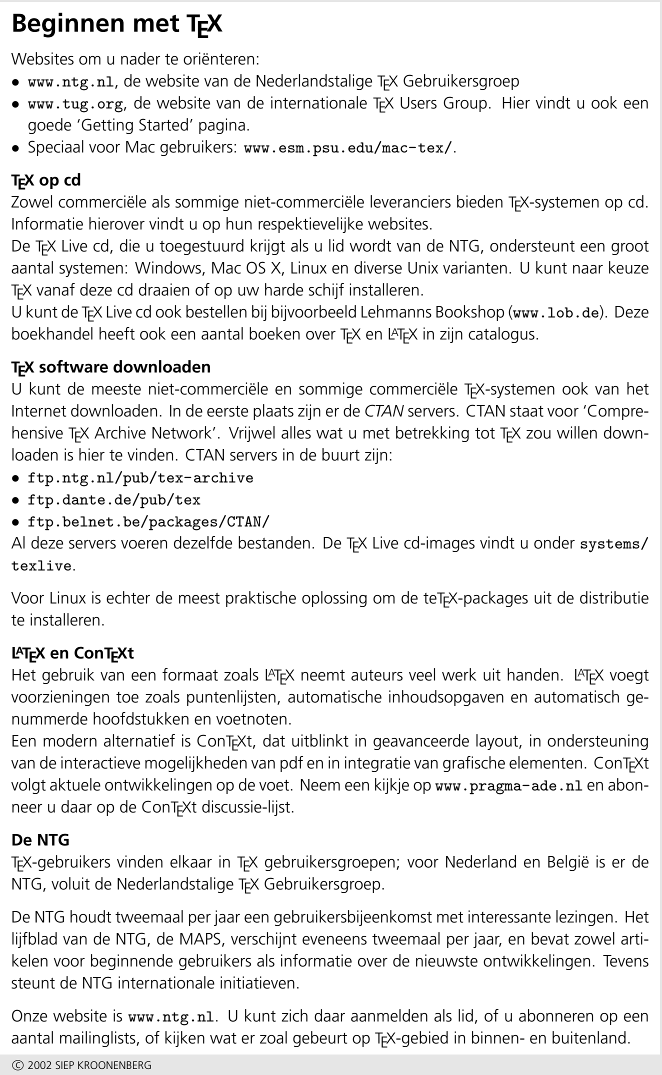
xypic



Lilypond



MetaPost



Beginnen met TeX

Websites om u nader te oriënteren:

- `www.ntg.nl`, de website van de Nederlandstalige TeX Gebruikersgroep
- `www.tug.org`, de website van de internationale TeX Users Group. Hier vindt u ook een goede 'Getting Started' pagina.
- Speciaal voor Mac gebruikers: `www.esm.psu.edu/mac-tex/`.

TeX op cd

Zowel commerciële als sommige niet-commerciële leveranciers bieden TeX-systemen op cd. Informatie hierover vindt u op hun respektievelijke websites.
De TeX Live cd, die u toegestuurd krijgt als u lid wordt van de NTG, ondersteunt een groot aantal systemen: Windows, Mac OS X, Linux en diverse Unix varianten. U kunt naar keuze TeX vanaf deze cd draaien of op uw harde schijf installeren.
U kunt de TeX Live cd ook bestellen bij bijvoorbeeld Lehmanns Bookshop (`www.lob.de`). Deze boekhandel heeft ook een aantal boeken over TeX en LaTeX in zijn catalogus.

TeX software downloaden

U kunt de meeste niet-commerciële en sommige commerciële TeX-systemen ook van het Internet downloaden. In de eerste plaats zijn er de *CTAN* servers. CTAN staat voor 'Comprehensive TeX Archive Network'. Vrijwel alles wat u met betrekking tot TeX zou willen downloaden is hier te vinden. CTAN servers in de buurt zijn:

- `ftp.ntg.nl/pub/tex-archive`
- `ftp.dante.de/pub/tex`
- `ftp.belnet.be/packages/CTAN/`

Al deze servers voeren dezelfde bestanden. De TeX Live cd-images vindt u onder `systems/texlive`.

Voor Linux is echter de meest praktische oplossing om de teTeX-packages uit de distributie te installeren.

LaTeX en ConTeXt

Het gebruik van een formaat zoals LaTeX neemt auteurs veel werk uit handen. LaTeX voegt voorzieningen toe zoals puntenlijsten, automatische inhoudsopgaven en automatisch genummerde hoofdstukken en voetnoten.
Een modern alternatief is ConTeXt, dat uitblinkt in geavanceerde layout, in ondersteuning van de interactieve mogelijkheden van pdf en in integratie van grafische elementen. ConTeXt volgt aktuele ontwikkelingen op de voet. Neem een kijkje op `www.pragma-ade.nl` en abonneer u daar op de ConTeXt discussie-lijst.

De NTG

TeX-gebruikers vinden elkaar in TeX gebruikersgroepen; voor Nederland en België is er de NTG, voluit de Nederlandstalige TeX Gebruikersgroep.

De NTG houdt tweemaal per jaar een gebruikersbijeenkomst met interessante lezingen. Het lijfblad van de NTG, de MAPS, verschijnt eveneens tweemaal per jaar, en bevat zowel artikelen voor beginnende gebruikers als informatie over de nieuwste ontwikkelingen. Tevens steunt de NTG internationale initiatieven.

Onze website is `www.ntg.nl`. U kunt zich daar aanmelden als lid, of u abonneren op een aantal mailinglists, of kijken wat er zoal gebeurt op TeX-gebied in binnen- en buitenland.

Nieuws van CTAN

Een uittreksel uit de recente bijdragen in het CTAN archief

Piet van Oostrum

abstract
Dit artikel beschrijft een aantal recente bijdragen uit het CTAN archief. De selectie is gebaseerd op wat ik zelf interessant vind en wat ik denk dat voor veel anderen interessant is. Het is dus wel een persoonlijke keuze. Het heeft niet de bedoeling om een volledig overzicht te geven. De uitgebreidere bijdragen zijn ook geen handleidingen. Beschouw het maar als een soort menukaart die de bedoeling heeft om de lezer nieuwsgierig te laten worden.

keywords
TEX, LATEX, packages, CTAN, classes, memoir, KOMA script, soul, underline, ulem, mathpazo, fonts.

LATEX classes

Het meest gebruikte TEX macropakket is nog steeds LATEX. Binnen de TEX wereld is de *de facto* standaard voor het maken van documenten van diverse aard, hoewel langzamerhand conTEXt ook meer terrein wint. Een belangrijke reden voor het gebruik van LATEX is dat verschillende uitgevers, tijdschriften en instellingen standaard classes hebben die hun specifieke stijl implementeren. Voor het overige gebruiken veel mensen de standaard LATEX classes `article`, `report` of `book` al dan niet opgetuigd met eigen definities of packages.

De structuur van LATEX is een kern waarin de gebruikelijke commando's zijn gedefinieerd, zoals de sectiecommando's, lijsten, referenties, etc. en een class file waarin de uiteindelijke implementatie van deze constructies gerealiseerd wordt, in het bijzonder de gewenste layout. Veel mensen associëren LATEX meer met het laatste dan met het eerste, en de typische "LATEX layout" is in feite de implementatie van de standaard bijgeleverde classes, en niet zozeer het eerstgenoemde framework.

De standaard classes zijn echter niet zo geweldig, misschien omdat er uiteindelijk niet zoveel aandacht is besteed aan een goede vormgeving, misschien omdat de vormgeving te Amerikaans is voor onze Europese ogen. Bovendien zijn deze classes niet erg parametriseerbaar waardoor het moeilijk is om de layout te veranderen. Er zijn dan ook diverse packages verschenen die als belangrijkste doel hebben om elementen van deze layout te veranderen. Hierbij kunnen we bijvoorbeeld denken aan `fancyhdr` voor het parametriseren van pagina headers en footers, `sectsty` en `titlesec` voor het veranderen van de layout van hoofdstuken sectietitels, `subfigure` voor het layouten van samengestelde figuren en `geometry` voor het parametriseren van de pagina layout.

Het combineren van een bestaande class file met een verzameling packages kan in het algemeen tot een min of meer bevredigend resultaat leiden maar er kunnen ook onderlinge conflicten tussen de packages zijn waardoor een en ander niet lukt of tot een ongewenst resultaat leidt. Overigens is een dergelijke *modulaire* aanpak in de informatica heel gebruikelijk en wordt in vele takken ervan ook gepropageerd, maar de structuur van LATEX (en waarschijnlijk zelfs de onderliggende structuur van TEX) is hier niet altijd even geschikt voor.

Een andere mogelijkheid is het schrijven van een eigen class, maar voor de meeste mensen is dit een te grote inspanning, temeer daar de documentatie over dit proces veel te wensen overlaat. Alleen uitgevers, tijdschriften en instituten zijn in het algemeen bereid om de benodigde inspanning te leveren.

De aanpak van conTEXt staat eigenlijk diametraal tegenover die van LATEX: conTEXt is zeer parametriseerbaar en wordt geleverd als een monolitisch pakket met weinig modulariteit. Het voordeel ervan is dat het intern consistent is en dat verschillende modules elkaar niet in de weg kunnen zitten. Dit wordt natuurlijk ook veroorzaakt door het feit dat de ontwikkeling grotendeels in één hand is. Bij een niet-modulaire aanpak is een grote parametriseerbaarheid natuurlijk ook gewenst omdat er anders geen mogelijkheid is voor persoonlijke aanpassingen (*customizing*).

Ik zal in dit artikel twee LATEX classes behandelen die proberen de nadelen van de standaard classes te overwinnen

Memoir

`Memoir` is een vrij recent verschenen class, geschreven door Peter Wilson. Zoals Wilson het zegt is het bedoeld voor "works of fiction and mathematical books", in feite voor de meeste soorten documenten waar LaTeX voor gebruikt wordt. Memoir kan beschouwd worden als een integratie van de LaTeX classes `article`, `report` en `book`, waarbij een grote mate van parametriseerbaarheid is ingebouwd die het gebruik van allerlei extra packages overbodig maakt.

De parametriseerbaarheid omvat aspecten als

- paginagrootte, zowel het papierformaat als de bladspiegel
- headers en footers
- hoofdstuk- en sectietitels
- alinea's, lijsten, citaties
- inhoudsopgave, lijst van figuren, etc.
- floating structuren
- bibliografie, index, etc.
- poëzie

De handleiding van `memoir` bevat meer dan 200 pagina's en begint met een introductie over grafische vormgeving. Het tweede deel is een beschrijving van de `memoir` parameters en commando's en vertelt ook hoe de vormgevingsprincipes uit het eerste deel met behulp van `memoir` geïmplementeerd kunnen worden. Het is natuurlijk een grote handleiding om door te lezen, maar omdat de basiscommando's die van de traditionele LaTeX classes zijn kan met eenvoudig beginnen en alleen die aspecten opzoeken die men wil veranderen.

`Memoir` kan ik van harte aanbevelen voor diegenen die niet tevreden zijn met de standaard classes van LaTeX en die zelf de vormgeving willen veranderen. Je kunt `memoir` vinden op CTAN in `macros/latex/contrib/supported/memoir`.

KOMA SCRIPT

KOMA SCRIPT is een verzameling classes die bedoeld is om de standaard LaTeX classes te vervangen. KOMA SCRIPT bestaat al langer, maar recent is een nieuwe versie uitgekomen, en bovendien is er nu naast de Duitse handleiding nu ook een Engelse beschikbaar waardoor deze classes voor een groter publiek toegankelijk zijn. De classes zijn oorspronkelijk ontworpen met het doel een wat meer Europese vormgeving ter beschikking te stellen.

De classes die in KOMA SCRIPT geleverd worden zijn:

`scrartcl`	vervangt	`article`
`scrreprt`	vervangt	`report`
`scrbook`	vervangt	`book`
`scrlettr`	vervangt	`letter` (Duits)
`scrlttr2`	vervangt	`letter`

De parametriseringsmogelijkheden van KOMA SCRIPT zijn minder uitgebreid dan die van `memoir` en bevatten o.a.

- papierformaat en pagina-layout
- headers en footers
- Hoofdstuktitels
- fonts

KOMA SCRIPT kan gevonden worden op CTAN in `/macros/latex/contrib/supported/koma-script`.

Soul

`Soul` is een package dat men kan gebruiken om stukken tekst te onderstrepen, doorstrepen, highlighten (van een achtergrondkleur voorzien) en voor letterspatiëring.

Onderstrepen en letterspatiëring worden in het algemeen als typografisch ongewenst beschouwd. Het zijn namelijk technieken die stammen uit het typemachinetijdperk, toen onderstrepen bijvoorbeeld cursief moest vervangen. Toch kan het voorkomen dat men in bijzondere gevallen deze technieken wil toepassen, bijvoorbeeld bij manuscripten (dus niet de uiteindelijke documenten), of bij speciale technische toepassingen. Doorstrepen kan bijvoorbeeld nuttig zijn om wijzigingen in een document aan te geven. Onderstrepen kan men met `\underline` maar dit heeft het nadeel dat de tekst niet afgebroken wordt. Het `soul` package heeft nu commando's voor deze speciale vormen die de tekst wel correct afbreken.

Er is ook een ouder package `ulem` dat ongeveer dezelfde functies heeft hoewel deze twee packages elkaar niet helemaal dekken. `Soul` is op CTAN te vinden in `macros/latex/contrib/supported/soul`.

Mathpazo

Een van de sterke punten van TeX is de ondersteuning van mathematische formules. Daarvoor heeft TeX een uitgebreide verzameling mathematische symbolen in de verschillende Computer Modern fonts. Helaaas zijn soortgelijke fonts in andere ontwerpen moeilijk te vinden. Als men bijvoorbeeld de broodtekst in Times Roman of Palatino wil zetten dan wordt vaak voor de formules gewoon Computer Modern gebruikt. Deze combinatie is echter niet aan te bevelen omdat de stijl van deze fonts veel te veel van elkaar verschilt. Voor Times Roman is het mogelijk om bijbehorende mathematische fonts te kopen. Als men het gratis wilt houden dan is het mogelijk om het package `mathptmx` te gebruiken. Dit selecteert zoveel mogelijk symbolen uit het Symbol font en de rest uit Computer Modern. Dit gebeurt met behulp van virtuele fonts. Overigens is dit nog niet ideaal, omdat er nog steeds ongelijkwaardige fonts gecombineerd worden.

Voor gebruik samen met het Palatino font is er nu de PazoMath familie, bestaande uit 5 fonts: PazoMath, PazoMath-Italic, PazoMath-Bold, PazoMath-BoldItalic en PazoMathBlackboardBold. Deze zijn specifiek ontworpen om goed bij Palatino te passen. De fonts komen in Type 1 formaat en bevatten o.a. Griekse hoofdletters (rechtop en schuin), kleine Griekse letters (schuin) zowel gewoon als vet, diverse mathematische symbolen, het Euro symbool en de “blackboard bold” symbolen.

Dit is tekst in Palatino met een € en hier de bijpassende formules:

$$E = mc^2$$

$$f(t) = \sum_{i=0}^{n} \gamma(x) e^{2\pi n t}$$

Blackboard bold: $\mathbb{C}, \mathbb{I}, \mathbb{N}, \mathbb{Q}, \mathbb{R}, \mathbb{Z}$

Het LaTeX package mathpazo.sty is bedoeld om een document met de Palatino en PazoMath fonts te zetten in plaats van de standaard Computer Modern fonts. Mathpazo is tegenwoordig onderdeel van het PSNFSS pakket dat standaard met alle LaTeX installaties meegeleverd wordt. Oudere installaties hebben dit echter nog niet aan boord. TeXlive versie 7 echter wel.

De fonts kunnen gevonden worden op CTAN in `fonts/mathpazo` en PSNFSS in `macros/latex/required/psnfss`.

Andere bijdragen

Onderstaande bijdragen zijn LaTeX packages of classes, tenzij anders vermeld. De locaties zijn op CTAN.

Circuit macros `graphics/circuit_macros` macros om diagrammen en elektronische circuits te tekenen

booktabs `macros/latex/contrib/supported/booktabs` tabellen in publicatiekwaliteit (beter dan de standaard tabellen)

musixps `fonts/musixtex/ps-type1` fonts in Type 1 formaat voor MusiXTeX (muziek zetten)

t1-fraktur `fonts/t1-fraktur` Fraktur Type 1 fonts

geometry `macros/latex/contrib/supported/geometry` definieer paginagrootte en layout

Eurosym `fonts/eurosym` font met het Euro-teken

placeins `macros/latex/contrib/other/misc/placeins.sty` macros om de beweging van floats te beperken

AstroSym `fonts/astro` astronomische symbolen

xmltex `macros/xmltex` macros voor het typesetten van XML (met of zonder LaTeX)

tocloft `macros/latex/contrib/supported/tocloft` parametriseren van de layout van inhoudsopgave, lijst van figuren en tabellen of eigen ‘lijst van’ constructies

euro-ce `fonts/euro-ce` Metafont sources voor Euro en ‘CE’ symbolen

thumbpdf `support/thumbpdf` programma om ‘thumbnails’ voor PDF te maken

fontinst `fonts/utilities/fontinst` programma om de benodigde files voor het gebruik van Type 1 fonts in TeX te maken

ttf2tex `support/ttf2tex` programma om de benodigde files voor het gebruik van Truetype fonts in TeX te maken

cmbright `fonts/cmbright` sans serif font familie, afgeleid van Computer Modern

manuscript `macros/latex/contrib/supported/manuscript` emuleert een document getypt op een typemachine

piechartMP `graphics/metapost/macros/piechartmp` Metapost macros om taartdiagrammen te maken

crop `macros/latex/contrib/supported/crop` voegt ‘cropmarks’ toe aan pagina

SQLTeX `support/SQLTeX` programma (preprocessor) om SQL statements in een LaTeX document te vervangen door hun output. Werkt met MySQL.

multicap `macros/latex/contrib/supported/multicap` ondersteunt captions binnen een multicol omgeving

cases `macros/latex/contrib/other/misc/cases.sty` package voor het formatteren van formules met cases

dtxtut `info/dtxtut` tutorial over de constructie van latex packages

ntheorem `macros/latex/contrib/supported/ntheorem` formattering van stellingen

mathabx `fonts/mathabx` font met extra mathematische symbolen

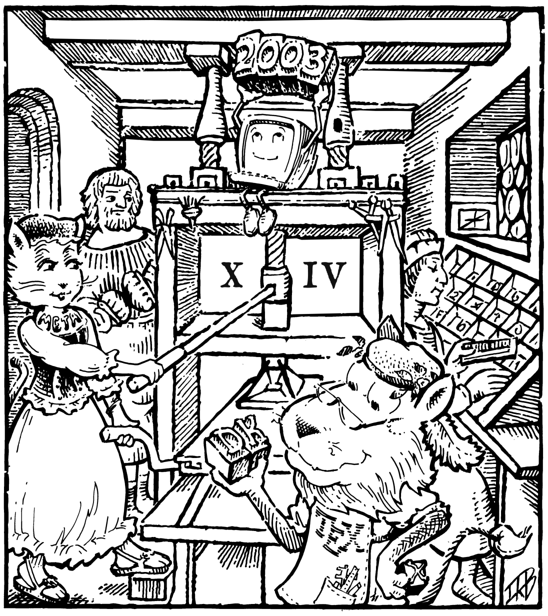
songbook `macros/latex/contrib/supported/songbook` macros voor liedboek (woorden en accoorden)

14th European TEX Conference

ELECTRONIC DOCUMENT ❧ DIGITAL TYPOGRAPHY ❧ TYPOGRAPHICAL HERITAGE AND ITS SURVIVAL IN THE COMPUTER ERA ❧ NEW TECHNOLOGIES APPLIED TO EDUCATION ❧ BOOK DESIGN AND PRODUCTION PROCESS ❧ FONTS ❧ DOCUMENT INTERNATIONALIZATION AND EXCHANGE

Deadline for abstract submission: January 7, 2003
Organizer: Yannis Haralambous, École Nationale Supérieure
des Télécommunications de Bretagne

Back to Typography



June 24-27, 2003
ENST de Bretagne
Brest Brittany
France

Programme Committee:

Jacques André (IRISA)
Hans Hagen (PRAGMA)
Yannis Haralambous (ENST-BRETAGNE)
Alan Hoenig (CUNY)
Ronan Keryell (ENST-BRETAGNE)
Frank Mittelbach (EDS)
John Plaice (UNSW)
Sebastian Rahtz (OUS)
David Salomon (CSU)
Petr Sojka (MUB)

Contact:

`http://omega.enstb.org/eurotex2003`
`http://www.enst-bretagne.fr`

ENST Bretagne

TUG 2003: The Silver Anniversary – 25 years! – of TEX

The 24th Annual Meeting and Conference of the TEX Users Group

tug2003@tug.org

Themes

- □ TEX Retrospective and Resurgence
- □ TEX–XML Symbiosis, TEI, Digital Archiving
- □ Fonts & Graphics
- □ pdfTEX, ConTEXt, Metapost, Metafont
- □ Installations & Management, CTAN
- □ Publisher and Prepress Dilemas
- □ New Developments & Directions
- □ MacOS X TeX: New Kid on the Block

Important Dates

2002	
Abstracts due	18 November
Abstracts accepted	18 December
Early-lion Registration	31 December

2003	
First draft of paper due	9 February
Registration Deadline	9 April
Final paper due	9 June

Links

TUG 2003 Homepage	http://wwww.tug.org/tug2003
Registration / Donations	https://tug.org/registration.html
Call for Papers	http://www.tug.org/tug2003/callfor.html
Heritage: TEX Retrospective	http://www.tug.org/tug2003/heritage/
TUG 2003 News Mailing List	http://www.tug.org/tug2003/news/

TEX Enthusiasts worldwide are invited to join us for a grand reunion
to celebrate the accomplishments of TEX,

July 20-24, 2003, Outrigger Waikoloa Beach Resort, Big Island, Hawaii tug

TUG 2002, Thiruvananthapuram

Michael A. Guravage

This past spring at BachoTEX Kaveh Bazargan made a presentation for TUG 2002 to be held this year in India; the first time a TUG conference has been held outside North America or Europe. As he spoke I smiled and conjured up images of far off Asia, but my mind said, "Impossible." Well, as time went on, I found myself musing more and more on the upcoming conference and India, then I recalled a few lines from a book written by G.K.Chesterton entitled Orthodoxy:

> "What could be more delightful than to have in the same few minutes all the fascinating terrors of going abroad combined with all the humane security of coming home again? . . . How can we contrive to be at once astonished at the world and yet at home in it?"

The thought of going to far off and exotic India with a group of people I know and whose company I enjoy, and to be immersed in TEX, was becoming irresistible. I quickly succumbed to this unique opportunity and began making plans. Thankfully, I found traveling companions in Hans Hagen and Fabrice Popineau. Hans and my trip began when we flew from Amsterdam to Paris; there we met Fabrice for the long-haul flight to Comumbo Sri Lanka, and then the final leg to Trivandrum India. During the flight Both Hans and Fabrice worked on their presentations, while I was confirmed as an unrehabilitated movie addict.

After our long flight and a wait at the luggage carousel where, for a moment, we wondered whether our bags had arrived, it was a pleasure to see Kaveh's smiling face as we left the airport. Trivandrum, or Thiruvananthapuram, is the capital of Kerala State in South India. Built on seven hills, it name means, "city of the sacred snake" named after a hydra like serpent from Hindu mythology.

We, along with nearly all the other foreign guests, stayed at the Hotel Samudra in Kovalam some 17 kilometers south of Trivandrum, a half hour drive from the airport. Kovalam consists of three small crescent shaped beaches on the Arabian sea. The hotel, run by the Kerala Tourist Development Commission, is located on a hill overlooking one of these idyllic beaches.

The Indian TEX Users group, The Indian chapter of the Free Software Foundation, and the Kerala State Department of Information Technology organized conference, whose venue was a high tech office complex called Technopark located slightly north of Trivandrum in Koryavattom. Every detail of the conference was meticulously organized; for example, air conditioned bus was put at our disposal for the forty minute commute from our hotel to the conference venue and back everyday. For those unfamiliar with Indian traffic, each trip made us grateful we were not in the driver's seat.

For a tourist, driving in India is a truly terrifying experience. Image a narrow two lane highway where the center line is a mere suggestion, and on which pedestrians, bicycles, bullock carts, various household pets, motorcycles, rickshaws, autos, lorries, busses, and the odd elephant all vie for position. The center line becomes a de facto passing lane, which means that passing vehicles are separated by a mere hand's breath. As a result, automobiles have no side view mirrors, or, if they do, they are neatly folded in. To announce his presence, and his intention to pass, a driver toots his horn. I am convinced that Indians replace their horns more often than their tires. Still, for all the chaos and clamor, we never saw a traffic jam, a irate motorist, or an accident.

Tutorials

Once at Technopark, the 63 delegates from 13 countries, settled down to a program of tutorials and presentations. Four tutorials given in the two days preceding the conference proper. On Sunday C.V. Radhakrishnan, Kaveh Bazargan's partner in Focal Image India, conducted an introduction to TEX and LATEX. Monday morning C.V. Radhakrishnan spoke again, this time describing Focal Image's document work flow for typesetting LATEX documents. Monday afternoon Hans Hagen traced METAPOST's history and its place in the TEX family tree and, with help from examples from his METAFUN manual, showed how METAPOST graphics can easily be included in ConTEXt documents. Tuesday, Sebastian Rahtz and Lou Bernard presented a full day tutorial on the Text Encoding Initiative (TEI). In the morning they showed how encoding, i.e. markup, makes a text's implicit structure and content explicit. This encoding is later used to analyze the text to provide multiple readings. Interestingly, a TEI DTD is a hierarchy of core, base, and auxiliary modules which together comprise a encoding tailored to a specific task – see their pizza metaphor for composing a DTD at `<www.tei-c.org/pizza.html>`. In the afternoon we surveyed the XSL family of tools including: XSLT, XPATH, and XSL FO.

Day One – Wednesday

The conference proper began Wednesday morning. An elephant sporting a TEX banner and a musical ensemble of traditional horns and drums called a *panchavadyam* greeted participants at the conference venue. Satish Babu, head of the Organizing Committee, chaired the opening session. Short addresses were given by two major conference sponsors: Ajay Shah from the Indian Ministry of Finance and Dr. K.R.Srivathsan from the International Institute for Information Technology and Management. The first keynote speaker, Ajit Ranade, a chief economist at ANB AMRO, presented an overview of economic climate in India, and a detailed look into the Indian software industry and its consequences for TEX. He estimates that 8000 people in India currently earn their living using TEX. S. Rajkumar from Linuxense Information Systems followed with a paper entitled, 'Indic typesetting – Challenges and Opportunities' where in he described the intrinsic complexity of Indic scripts, and the special processing necessary to transform them from Unicode text to font metrics TEX can use. Of the 5000 commercial Indic fonts available, only 20 are available in TEX. Next, Amitabh Trehan from the Mahatma Gandhi Antarrashtriya Hindi Vishwavidyalaya in Delhi presented a paper entitled 'Typesetting in Hindi, Sanskrit and Persian.' He suggested that the future of TEX is bound to its ability to typeset multiple languages. As proof, he showed a book of Indian verse completely typeset in LATEX using the devnag, sanskrit, and ArabTEX packages. Continuing her work first presented at BachoTEX earlier this year, Gyöngyi Bujdosó told delegates about her continuing work to instruct LATEX in the subtleties of Hungarian typographical conventions. After lunch Satish Babu from the Computer Society of India spoke about, 'New Horizons of Free Software.' India's insatiable appetite for software, her populous and well educated work force, and her relatively low per capita income combine to make Free Software an essential model for software development on the subcontinent. With 'The Tao of Fonts' Wlodzimierz Bzyl combined METAFONT and type 3 fonts to dazzle us with a variety of unique letter shapes – including hexagrams from the Chinese book of changes (I Ching). The first day ended with Roozbeh Pournader from the Sharif University of Technology in Tehran presenting a paper entitled, 'Unicode, the Moving Target.' He began by describing the Unicode standard and several recently introduced features, he went on to enumerate several new requirements for Omega, and concluded by reiterating how important standards are.

Day Two – Thursday

The keynote speaker Thursday morning was Hans Hagen who, in his talk entitled 'ConTEXt XML and TEX', spoke about the current expectations of educational publishers and DTP. With an example from the journal of the Dutch Mathematical Society, Hans showed how ConTEXt supports XML driven document workflows that produces high quality output to rival or exceed that produced by commercial DTP systems. In a talk entitled, 'Revisiting WYSIWYG Paradigms for Authoring LATEX', Kavid Kastrup compared five WYSIWYG-like TEX editing systems for ease of use and fidelity of visual feedback. Next, Ross More from Macquarie University in Sydney spoke about 'serendiPDF with searchable math-fields in PDF documents.' By storing the LATEX source for a math environment in a hidden field inside a PDF document, Acrobat Reader can be persuaded to display the LATEX source, which can be copied and pasted into other documents. Stephen Watt from the University of Western Ontario followed with his paper entitled, 'Conserving Implicit Mathematical Semantics in Conversion between TEX and MathML.' High level semantics are lost in the standard translation procedure that recursively expands every macro before translation. Alternatively, translations based on an isomorphic mapping between higher order structures, e.g. TEX macros and XLST templates, can preserve the mathematical semantics. Some may remember Karel Piska's talk at BacoTEX where he compared CM and EC Type 1 fonts. This afternoon Karel, from the Czech Academy of Sciences Institute of Physics, presented his work entitled, 'A Conversion of Public Indic Fonts from METAFONT into Type 1 Format with TEXTRACE.' Taking the METAFONT sources for some 55 Indic fonts, he corrected and simplified the resulting outline paths to produce scalable versions for inclusion in PDF documents. Next, Behdad Esfahbod from the Sharif University of Technology in Tehran spoke about 'FarsiTEX and the Iranian TEX Community.' The advent of PDF has induced the FarsiTEX project team to include PostScript Type 1 fonts in the newest release. The Persian character set and its inherent semantics make a special text editor an essential part of the FarsiTEX system. However, supporting the bidirectional character of Persian in the editor remains problematic. Dennis Roegel from LORIA France presented, 'MetaObj': Very High-Level Objects in METAPOST.' He explained the structure and use of MetaObj's standard object classes which includes, among other things, boxes, containers, and trees.

Karel Skoupy from ZTH in Zurich ended the day's lectures with his proposal for a, 'New Typesetting Language and System Architecture.' Besides supporting multiple input and output formats, a new 'modular' architecture should have a single model for processing both text and graphics, while separating language support from the typesetting engine.

Trivandrum Town

Friday's conference schedule was interrupted by the threat of a general strike called by local trade unions in protest of the local government's proposed doubling of energy tariffs. Since it was inadvisable to drive from the hotels to Technopark that day, the conference organizers tweaked the program schedule, leaving Friday free. Those who stayed at Hotel Samudra engaged in various TEX related discussions, while David Kastrup gave a tutorial on the intricacies of TEX macro expansion that he was unable to give earlier in the week.

However, at the eleventh hour the government capitulated, and the strike was called of. So, when the conference organizers decided to stick with the revised schedule, Steve Grathwol of Duke University Press and I set off for Trivandrum Town together in a three wheeled rickshaw.

This was my first visit to Asia, and though I had read about what to expect, the abstract never matches the actual when traveling. In Trivandrum the market is confined with several forts, or gates, that define the town's center. Imagine center of Amsterdam or Utrecht, not filled with shops, but with kraampjes – a perpetual Koninginnedag. In one shop someone is sewing on a single foot powered sewing machine. The next shop's display of fresh fruit and roasted nuts is a rapture for my eyes and nose. The hammering from the next shop is someone trying to repair a rickshaw's frozen brake drum. And there are people everywhere: school children in uniform, beggars, businessmen with cell phones, university students with books, housewives with groceries, manual laborers, street vendors, and loiters – a city of more than a million souls, and I had no idea where I could buy a pair of double A batteries.

Trivandrum is a bustling city where every neighborhood seems to be a self contained village. The people we met were generous to a fault. On one occasion I was invited to the home of one of the conference's audio-visual technicians. When we arrived at his home, I was warmly welcomed by he and his family. Later that evening, after a spicy dinner at a local restaurant, he refused to let me contribute a cent to the cost of the meal or the petrol he used taking me back to the hotel.

We in the west are proud of our free market economies, but in India we encountered pure price discrimination. Whether about silks or suitcases, spices or sea shells, the familiar question, 'how much does it cost?' is there transformed into, 'how much am I willing to pay for that commodity or service?' For example, a rickshaw ride across town in the company of an Indian host cost 7 Rupees, or 3$^1/_2$ cents; the same trip the next day cost me 50 Rupees, a mere Euro. I was happy to pay the fare, even though it was seven times the local rate. Such encounters are inevitable when disparities in income are apparent.

Day Three – Saturday

The conference resumed Saturday morning the TUG business meeting first on the agenda. Ross Moore read the minutes from the Board of Directors meeting held in Portland Oregon in July. Kaya Christiansen from the University of Århus in Denmark then described the new TEX Development Fund. Concern was raised over the late publication schedule for TUGBoat, and also at the revelation that TUG membership has fallen by 14% in each of the last two years. Finally, delegates were encouraged to put forward nominations for President and Board members for the upcoming election in 2003.

The technical program continued with G. Nagarjuna speaking about, 'The Semantic Web, GNOWSYS, and Online Publishing.' He was followed by Dr. K.R. Srivathsan who described an 'Education Grid' – a system for coordinating computer and educational resources for students in higher education. Next, Hong Feng, founder and Chairman of the Chinese TEX Users Group, described Lojban – an unambiguous artificial language used to express Chinese text encoding in readable ASCII characters. K. Anilkumar from Linuxense Information Systems in Trivandrum concluded the morning session with a presentation entitled, 'Databases, TEX and Online Complex Report Generation' where he demonstrated how shell-escapes such as `\write15`, `\write16`, `\write18`, and `\input` can be harnessed to read databases and generate reports. John Plaice from the University of New South Wales and co-author, along with Yannis Haralambous, of Omega, described the 'Low-level Devanagari Support for Omega.' John went on to mention the improvements planned for Omaga 2, e.g. better diacritics placement, and even ventured to speculate that Omaga 3 would incorporate a new object oriented typographical programming language, and depart from the traditional TEX typesetting model. Fabrice Popineau from SUPELEC in Metz told us all how 'TEXLive 7 under Windows has been improved with, among other things, extensions to Kpathsea to support URL syntax. Karl Skoupy made the last technical presentation of the conference with his proposal for a 'TEX File Server.' Multiple TEX runs is the usual practicing for placing graphics and resolving references. A TEX file server could save and reuse the data structures currently lost between separate runs of Kpathsea.

After an invitation by Ross Moore and Wendy McKay to join TUG 2003 in Hawaii, Dominic Wujastyk from University College London closed the conference by expressing everyones thanks to the conference organizers for their dedicated and tireless efforts in making TUG 2002 a success.

Cuisine & Entertainment

I would be remiss if I did not mention that the conference delegates enjoyed delicious buffets for lunch at Technopark and again for dinner at Hotel Samudra. The sun shone every day, and in the evening we ate on tables set out on the lawn under a clear sky of stars. Wednesday evening we were treated to a demonstration of *kalari payattu* – the traditional Kerala martial arts form. Dominik Wujastyk told the history of the art before the performance began; which included knives, swords, clubs, flaming staffs, and a particularly terrifying weapon which looked like a cross between a sword and a whip. The official conference dinner was held on Thursday evening; followed by a dance exhibition performed by children from the Sri Chitra home for the Poor and Destitute in Trivandrum, and a flute recital of classical Carnatic music by V. C. George.

The Fourth Estate

The conference appeared prominently in print media during our stay. The Hindu, The Economic Times, and the Business Express all ran articles reporting on the conference with titles like: 'Conference on TEX software', 'TEX targets large Indian user group', 'Free scientific publishing software set to hit Asia' in which Donald Knuth was described as a Info-tech guru. Later, The Hindu ran another lengthy article in their business section entitled, 'TEX: a free tex-processing tool.' With contributions from Dominik Wujastyk, Kaveh Bazargam, C.V. Radhakrishnan, and K.G. Kamar, you may rest assured that the article was well informed.

Backwater Tours

After the conference, several delegates engaged local tour operators for a cruise on the large lake and connecting waterways north of Trivandrum near the town of Alleppey and the coastal city of Cochin. Tourists can hire old rice barges that have been refurbished as wicker houseboats – replete with bedrooms, a dining room, and a sitting-room. A crew of 3 or 4 man the boot and prepare your meals.

When our boat docked in the evening, we quickly discovered that the surrounding boats were all engaged by TEXies. It appeared that the conference has merely taken ship. All those on our boat were part of a small group whose accommodations every step along the way were always addressed to a certain Mr. Hans. But that, my friends, is an other story.

Patrick Gundlach

erratum
meta-euro

The new currency, the euro, has not only brought new coins and bills to a lot of people in europe, but it has also introduced a new symbol, which looks like this: €. It is not just a capital C with two horizontal bars. There are angles at the left and right side of the bars as well as at the upper left corner of the C. Even the opening angle of the C should be the same in all the symbols. The european union (eu) tries to guide the font designers and provides an official version of the symbol on their webpage. It looks like a geometric construction that can be drawn with a ruler and a compass. And—since we are all used to solve such tasks with a computer—this can certainly be done with MetaPost. MetaPost is a companion to Don Knuth's METAFONT and outputs encapsulated postscript (eps) rather than fonts coded in bitmaps. MetaPost was written by John Hobby. It uses a language that is very close to the one described in the METAFONT book. MetaPost (as well as the changes to METAFONT) is described in the MetaPost user's manual (available as `mpman.ps` on ctan or your local TeX installation) It is advisable, if you plan to program in MetaPost, to have the users manual available as well as Knuth's METAFONT book. But the user's guide will probably suffice at the beginning. There is also a nice introduction to MetaPost in the MetaFun handbook. MetaFun is a macro package that is designed to work well with the TeX macro package ConTeXt. But beginners should be aware of the differences of plain MetaPost and the MetaFun extensions when reading the MetaFun manual.

But back to the euro symbol. I have stated that there is an official version provided by the eu, that looks as it can be constructed by geometrical means. It is depicted in figure 1.

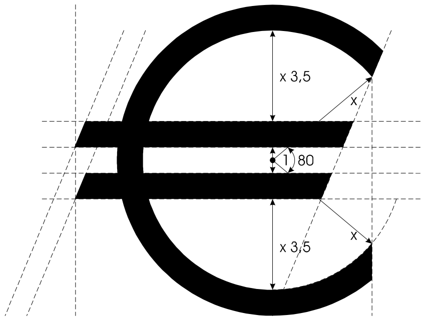


Figure 1

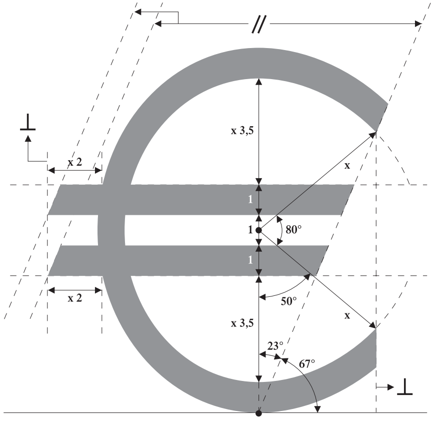


Figure 2

You will probably notice that there are some angles and some distances shown. But if you look close enough, there are also some sizes unknown. For example it is not clear how far the horizontal bars stick out to the left side. There is another construction by the european central bank (ecb). You can see their version in figure 2.

This construction is about the same as the one mentioned before, but it adds some additional values. You can see that the horizontal bars now stick out a well defined amount. Most other constructions in the media are more or less exact copies of these two. The eu one seems to be in favour. For the rest of this article that one will be used.

I have mentioned earlier that there are some weaknesses of the official construction, that makes it impossible to make an exact reproduction without measuring distances by hand. The problems are

1. the length of the horizontal bars is unknown
2. the shape of the surrounding “circle” is not clear. Only the height and the relative position of the horizontal bars is defined
3. the thickness of the “circle” is unknown
4. the thickness of the horizontal bars is unknown
5. and, not really a problem, the lables (x) on both angles are misleading and do not seem to fulfill a real purpose.

Perhaps these points are unclear on purpose, so the font designer has some freedom to let the euro symbol fit nicely into a given font. Whatever the reason is, we have to make assumptions about the unclear parts, in order to be able to express the shape in terms of a METAPOST program. I will, for the sake of clearness, assume that the surrounding shape is a circle. While it looks like it in the eu construction, it looks a bit more squeezed in the ecb one. In the official construction there is a variable x that seems to be some kind of a unit. The diameter of the circle is $11x$ if the horizontal bars and the circle have a width of $1x$.

I will now show the METAPOST code. See below for an explenation on how to use it in LATEX or ConTEXt.

```
prologues    := 2 ;
beginfig(1);

def tand expr x = (sind (x) / cosd (x)) enddef;

path unitcircle; unitcircle:=fullcircle scaled 2;
%unitcircle has radius of 1
```

Every METAPOST picture is enclosed in beginfig/endfig. The number in the parentheses denotes the file extension. Say, for example, you have named the METAPOST-file maps-euro.mp, the figure gets written as maps-euro.1. In plain METAPOST there is no function to calculate the tangens. Since it is needed below, I provide a definition here. I also provide a circle with radius 1. I could of course have used the (predefined) fullcircle and just multiply all values by two.

```
boolean show_dots, show_lines,show_labels ;

show_lines  := false;
show_dots   := false;
show_labels := false;
```

These lines are only for debugging purpose. The euro-symbol, when used in text, should not have any labels or lines. But while constructing the symbol I would like to see if the points are in the right place. And, of course, it should make it easier for the reader to follow my thoughts as expressed in this code. Just set these variables to true if you want to see more output.

```
show_lines  := true;
show_dots   := true;
show_labels := true;
```

What follows is declaration of some variables. In METAPOST you have to tell the system whether your variable is a path, a pair or ....

```
10  path inner, outer; % the big circle: inner and outer part
11  path hbar;         % horizontal bar
12  path a[];          % aux paths for intersections
13  path clippath;     % path for clipping
14  pair topbarleft, bottombarleft;
15  pair o[];          % official points
16  pair c[];          % clip points
17  picture cp;        % currentpicture for clipping

18  x :=1cm;
19  radius:=5.5x;
20  topbarlength := 10x;
21  thickness:=1x;
```

These are the only user-changable parameters I provide. In a typical MetaPost program, there are probably more values that can be influenced. x is our unit as given in the official construction. It determines the overall size of the symbol. But since the picture is scalable, it really does not matter too much to what value you set it.

Now I will determine the points that are needed to draw the symbol. In the official construction, the points o1, o2 and o3 are given, as well as the right slant. Alpha is the angle between the horizontal line and the right slant.

```
22  o1 = (0 ,-radius-.5thickness);
23  o2 = dir  40 * (radius-.5thickness);
24  o3 = dir -40 * (radius-.5thickness);
25  alpha := angle (o2-o1);

26  a1 := o1 -- o2;      % the right slant
27  a2 := origin -- o3;  % the lower 40deg. angle
```

Now, let us actually draw something. Start with the circle. This circle is a fullcircle. The unwanted part at the right side will be clipped off later. After drawing the circle, we determine the shape of the horizontal bars. There again, we draw more than we need and clip the unwanted part off later.

```
28  draw unitcircle scaled radius
29        withpen pencircle scaled thickness;

30  hbar := unitsquare xscaled topbarlength yscaled thickness
31                 slanted (1/tand (alpha));

32  % top bar lrcorner:
33  c3 := ((-infinity,0.5x) -- (infinity , 0.5x)) intersectionpoint a1;

34  topbarleft    := (xpart c3 - topbarlength , 0.5x);
35  bottombarleft := (xpart c3 - topbarlength ,-0.5x - thickness);

36  fill hbar shifted topbarleft;
37  fill hbar shifted bottombarleft;
```

In line 33 we have found c3 just by asking MetaPost to find the intersection of the path a1 and an infinite long horizontal line shifted 0.5 units above the origin. We do not assign this line to a variable or even draw it. Now the shape of the horizontal bars are known, so we can draw them.

At this time everything needed is drawn. But since we have put more in the picture than necessary, some erasing must be done. There are several ways to accomplish this. You can overdraw the unwanted parts with the background color. But since the background color is not known at this time, we cannot use this method. Another solution is to clip the picture to a certain path. This is as if you take a pair of scissors and cut along the path. We use clipping now:

```
cp := currentpicture;

c2 := a1  intersectionpoint a2;
c4 := o2 + dir alpha *2thickness;
c5 := (xpart  o3, ypart lrcorner cp);

outer := unitcircle scaled (radius+.5thickness);
c1 := (o1--c4) intersectionpoint outer;

clippath := llcorner cp -- c5 -- o3 -- c2 -- c1 --
          (xpart c1,ypart urcorner cp) -- ulcorner cp -- cycle ;

clip currentpicture to clippath;
```

What is being done here, is to find a path that can be used for clipping. It has to be as tight as possible to the symbol in order to avoid unnecessary space. Since the left side of the horizontal bars is as far as we have drawn yet, we can use the left boundary of the current picture (for example, `llcorner` denotes the lower left corner of a picture). c4 is not used for clipping, it is merely an auxiliary point to find c1.

That's it! With these few lines you draw the euro-symbol. But what is even more important, is that you have not merely presented the shape (this could be done with most drawing programs), you have expresed *how* to draw it. Contrary to the official version provided by the european union, there is no part in the construction unclear. We have even stated (although not derived from the official construction) which parts of the symbol may be changed at user option (the radius for example). This is a great advantage over using a simple (or not so simple) drawing program.

The rest of the program (except the `endfig` and `end` in the last two lines) is for debugging purpose. Remember the `show_line` (and the other variables) you have set either to true or false in the beginning? The `drawoptions` statement states a default for all following draw commands.

```
if show_lines:
  inner := unitcircle scaled (radius-.5thickness);
  pickup pencircle scaled 1pt;
  drawoptions (withcolor .7white);

  draw inner; draw outer;

  draw a1 dashed evenly;
  draw origin -- o2 dashed evenly;
  draw a2 dashed evenly;
  draw hbar shifted topbarleft;
  draw hbar shifted bottombarleft;
  draw clippath;
fi
```

```
59 if show_dots:
60  pickup pencircle scaled 2pt;
61  drawoptions (withcolor .5white);
62  draw origin;
63  draw c1; draw c2;
64  draw c3; draw c4;
65  draw c5;
66  draw o1; draw o2;
67  draw topbarleft;
68  draw bottombarleft;
69 fi
70 if show_labels:
71   defaultfont:="ptmr8r";
72   drawoptions (withcolor .5black);
73   label.bot("origin",origin);
74   label.bot("bottombarleft",bottombarleft);
75   label.bot("topbarleft",topbarleft);
76   label.rt ("c1",c1);
77   label.rt ("c2",c2);
78   label.rt ("c3",c3);
79   label.rt ("c4",c4);
80   label.bot("c5",c5);
81   label.bot("o1",o1);
82   label.bot("o2",o2);
83   label.rt ("o3",o3);
84 fi
85 if show_lines:
86    draw bbox currentpicture dashed evenly withcolor .7white ;
87 fi
88 endfig;
89 end;
```

The endfig is just the opposite of the beginfig. It denotes the end of a picture. The end instructs MetaPost to stop reading the input file and, in this case quit.

From this point on I will suppose that you have keyed in all the numbered lines into your editor and saved the file with the name maps-euro.mp. You can run MetaPost by typing the command

```
mpost maps-euro
```

into your favorite shell. MetaPost will generate a file called maps-euro.1. (For a description of the prologues command in line 1 see the MetaPost manual or the Latex-Graphics Companion.) This is a regular eps file, except when you have used fonts in your program. In our case if you turned on the show_labels switch. MetaPost does not include the fonts in the output file, it merely includes a reference in the eps file. So if you use a MetaPost picture with fonts, it is best to use it in conjuntion with TeX. The following simple Latex wrapper will suffice:

```
\documentclass{article}
\usepackage{graphicx}
\begin{document}
\includegraphics{maps-euro}
\end{document}
```

But this will work only if you have renamed maps-euro.1 to maps-euro.eps.

Using this symbol with ConTeXt is also possible. Just copy the lines starting after the beginfig and ending just before the endfig into a file, say maps-euro.tex. Enclose these lines in `\startuseMPgraphic{maps-euro}` and `\stopuseMPgraphic`. So you have (in maps-euro.tex)

```
\startuseMPgraphic{maps-euro}
def tand expr x = (sind (x) / cosd (x)) enddef;

path unitcircle; unitcircle:=fullcircle scaled 2;

...

if show_lines:
   draw bbox currentpicture dashed evenly withcolor .7white ;
fi
\stopuseMPgraphic
```

Actually you even do not need line 3 and 71, since tand is already defined in metafun (ConTeXt uses the MetaFun macro package when running MetaPost) and the default font is the current ConTeXt font. You can include the euro symbol as a picture by using `\useMPgraphic{maps-euro}` in your file or you can use it as 'character' that can be typeset in your text. It will be scaled to the current text size. What you have to do is write a small graphic-wrapper that allows scaling to the demanded height:

```
\startuniqueMPgraphic{euro}{height}
   \includeMPgraphic{maps-euro} ;
   currentpicture := currentpicture ysized \MPvar{height} ;
\stopuniqueMPgraphic
```

Now you can use this intermediate graphic to define a symbol:

```
\definesymbol[euro][\uniqueMPgraphic{euro}{height=1.5ex}]
```

To get a € you can use `\symbol[euro]` in your text.

Maps Editors

erratum

MathML Correction

In Hans Hagen’s article on MathML processing in CONTEXT, there were four example files missing due to an error in the production process. Here they are, with our apologies to Hans Hagen.

File pc-i-380.xml, processed, on page 69:

$$\int \frac{1}{\cos\left(ax\right) 1 \pm \sin\left(ax\right)}\,\mathrm{d}x = \mp \frac{1}{2a\, 1 \pm \sin\left(ax\right)} + \frac{1}{2a}\log\tan\left(\frac{\pi}{4} + \frac{ax}{2}\right)$$

File pc-i-380.xml, source, on page 69:

```
<math>
  <apply> <eq/>
    <apply> <int/>
      <bvar> <ci> x </ci> </bvar>
      <apply> <divide/>
        <cn> 1 </cn>
        <apply> <times/>
          <apply> <cos/>
            <apply> <times/>
              <ci> a </ci>
              <ci> x </ci>
            </apply>
          </apply>
          <apply> <fn> <ci> &plusminus; </ci> </fn>
            <cn> 1 </cn>
            <apply> <sin/>
              <apply> <times/>
                <ci> a </ci>
                <ci> x </ci>
              </apply>
            </apply>
          </apply>
        </apply>
      </apply>
    </apply>
    <apply> <plus/>
      <apply> <fn> <ci> &minusplus; </ci> </fn>
        <apply> <divide/>
          <cn> 1 </cn>
          <apply> <times/>
            <cn> 2 </cn>
            <ci> a </ci>
            <apply> <fn> <ci> &plusminus; </ci> </fn>
              <cn> 1 </cn>
              <apply> <sin/>
```

```
                <apply> <times/>
                  <ci> a </ci>
                  <ci> x </ci>
                </apply>
              </apply>
            </apply>
          </apply>
        </apply>
      </apply>
      <apply> <times/>
        <apply> <divide/>
          <cn> 1 </cn>
          <apply> <times/>
            <cn> 2 </cn>
            <ci> a </ci>
          </apply>
        </apply>
        <apply> <log/>
          <apply> <tan/>
            <apply> <plus/>
              <apply> <divide/>
                <ci> &pi; </ci>
                <cn> 4 </cn>
              </apply>
              <apply> <divide/>
                <apply> <times/>
                  <ci> a </ci>
                  <ci> x </ci>
                </apply>
                <cn> 2 </cn>
              </apply>
            </apply>
          </apply>
        </apply>
      </apply>
    </apply>
  </apply>
</math>
```

File pc-s-001.xml, processed, on page 80:

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots = \frac{\pi}{4}$$

File pc-s-001.xml, source, on page 80:

```
<math>
  <apply> <eq/>
    <apply> <plus/>
      <cn> 1 </cn>
      <apply> <minus/>
        <apply> <divide/>
          <cn> 1 </cn>
          <cn> 3 </cn>
        </apply>
      </apply>
      <apply> <divide/>
        <cn> 1 </cn>
        <cn> 5 </cn>
      </apply>
      <apply> <minus/>
        <apply> <divide/>
          <cn> 1 </cn>
          <cn> 7 </cn>
        </apply>
      </apply>
      <ci> &cdots; </ci>
    </apply>
    <apply> <divide/>
      <ci> &pi; </ci>
      <cn> 4 </cn>
    </apply>
 </apply>
</math>
```

File wh-g-001.xml, processed, on page 82:

$$\sin\left(x+y\right)=\sin x\cos y+\cos x\sin y$$

File wh-g-001.xml, source, on page 82:

```
<math>
  <apply> <eq/>
    <apply> <sin/>
      <apply> <plus/>
        <ci> x </ci>
        <ci> y </ci>
      </apply>
    </apply>
    <apply> <plus/>
      <apply> <times/>
        <apply> <sin/>
          <ci> x </ci>
        </apply>
        <apply> <cos/>
          <ci> y </ci>
        </apply>
      </apply>
      <apply> <times/>
        <apply> <cos/>
          <ci> x </ci>
        </apply>
        <apply> <sin/>
          <ci> y </ci>
        </apply>
      </apply>
    </apply>
  </apply>
</math>
```

File wh-m-002.xml, processed, on page 98:

$$|I| = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} = 1$$

File wh-m-002.xml, source, on page 98:

```
<math>
  <apply> <eq/>
    <apply> <determinant/>
     <ci> I </ci>
    </apply>
    <apply> <determinant/>
      <matrix>
        <matrixrow> <cn> 1 </cn> <cn> 0 </cn> </matrixrow>
        <matrixrow> <cn> 0 </cn> <cn> 1 </cn> </matrixrow>
      </matrix>
    </apply>
    <cn> 1 </cn>
  </apply>
</math>
```

fonts

Fonts for the MAPS

Siep Kroonenberg
siep@elvenkind.com

keywords
fonts, texnansi encoding, virtual fonts, reencoding

Ever since the redesign of the MAPS (actually, it wouldn't hurt to have another one by now), we have used Times, with real small-caps and old-style figures, for body text; Frutiger for headings and special items, and narrowed Courier as monospaced font. Under the hood, however, font support has been redone twice.

Encodings

TeX references characters by number, and expects certain characters at certain slots. This assignment of characters to slots is called an encoding. With macros, you can make TeX expect a different encoding.

TeX's native encoding has some peculiarities:

- it only uses the first 128 slots
- it doesn't contain accented characters; these are synthesized from base characters and accents
- the first eleven slots contain Greek capitals
- it doesn't contain the Polish Ł or Ł, but is *does* contain the slash of these characters as a separate character

OT1 is a slight modification of TeX's native encoding which is more suitable for commercial fonts.

TeX specialists don't like synthesized accented characters because they interfere with hyphenation. Therefore, this encoding is being superseded by 8-bits encodings which use most of the 256 available slots and include the more important accented characters outright.

The most widely used such encoding is T1. Unfortunately, it lacks a number of symbols such as the dagger- and copyright symbols. This necessitated the introduction of the text companion- or TS1 encoding. On top of that, some of the accented characters, which do not normally occur in Western European languages, don't occur in commercial Type 1 fonts. These characters have to be synthesized with the use of virtual fonts. Below, more about virtual fonts.

An alternative encoding is texnansi or LY1. It limits itself to characters present in commercial Type 1 fonts, but contains most of the characters from the TS1 encoding, and is sufficiently complete for typesetting Western European languages. It has been introduced by Y&Y with their commercial TeX distributions, which don't support virtual fonts at all. Y&Y offers support files for texnansi for free at their website `www.yandy.com`.

An aside: once we can move to Omega, Unicode and large character sets, these annoying issues should automagically disappear (or maybe get replaced with other annoying issues).

Virtual fonts

A virtual font is a recipe for dvi-drivers for synthesizing new fonts from other fonts. Such a recipe might create accented glyphs by placing accent glyphs above base characters, or selectively scale glyphs, or borrow some glyphs from other fonts.

TeX will only handle the resulting metrics, and won't care that they came from a virtual rather than a real font. Virtual accented characters are fine as far as TeX hyphenation goes.

The main problem with virtual fonts is that they add complexity to an already overly complex TeX installation. To use a font in T1 encoding, we need a tfm for the real font, e.g. ptmr8r.tfm, plus a tfm for the virtual font: ptmr8t.tfm, plus the recipe: ptmr8t.vf. Quite likely, we also need the text companion font, which means another two files: ptmr8c.tfm and ptmr8c.vf. In contrast, for texnansi encoding a single file ptmr8y.tfm would suffice.

Fontinst

Adobe has its own afm format for storing font metrics. Installing commercial Type 1 fonts is largely a matter of converting Adobe's afm font metrics to TeX's tfm font metrics.

TrueType fonts can also be used with TeX. Mainstream TeX distributions contain a number of utilities for handling TrueType fonts. For pdftex in particular, using TrueType fonts is not much different from using Type 1 fonts. Although below I'll write about Type 1 fonts, most of it is also relevant for TrueType.

The major free TeX distributions offer two ways to install commercial fonts in TeX: fontinst and afm2tfm. Both derive TeX font metrics from such an afm-file.

Fontinst *only* lets you use virtual fonts. There is a command `\latinfamily` that lets you install one font family at one fell swoop: 'raw' fonts, the old OT1 encoding, the newer T1 encoding, its companion TS1 encoding, all these with artificial small-caps and artificially slanted versions where [not totally in]appropriate. The complexity of the interface deters people from attempting more fine-grained control of fontinst, so using fontinst probably will add a lot of clutter to your system. On the plus side, fontinst also generates LATEX support files and mapfile fragments.

The initial MAPS font setup, first used for issue 19, contained OT1-encoded tfms generated with fontinst. It was necessary to delve into the low-level interface of fontinst to replace the regular figures of Times with old-style figures. Somehow, the real small-caps never got activated. This got fixed for MAPS 24, at least, for the LATEX articles. At that occasion, the encoding was changed from OT1 to T1 plus TS1.

afm2tfm and afm2pl

Afm2tfm, which is a companion program to dvips, doesn't do virtual fonts in its default mode. It simply converts one afm to one tfm, and writes a mapfile line to standard output. It does support slanting, widening/narrowing and reencoding. Unfortunately, for some reason it isn't designed to write kerning- or ligature information to the tfm in its default mode.

On the Y&Y site you can read how you can apply afm2tfm in a roundabout way and create tfms which *do* have their original kerns and ligatures.

Afm2pl is my own adaptation of afm2tfm. If you are interested: you can find it at `www.ntg.nl/afm2pl.html`. It does preserve kerns and ligatures. Instead of tfm files, it produces pl files, which are their human-editable ascii counterparts. You can convert back and forth between pl and tfm with the pltotf and tftopl utilities, which are part of any TEX installation.

Although afm2pl by itself can't do small-caps, it *can* create all-caps fonts, since that is simply a matter of reencoding. The distribution contains an encoding vector for this. The latest, source-only distribution optionally does letterspacing, which can make an all-caps font look much nicer.

For this MAPS issue, I used afm2pl to generate texnansi-encoded tfms. However, fontinst was needed to create Times fonts with old-style figures and to create artificial small-caps for the Frutiger family.

At some future date, Taco Hoekwater and I hope to put together a fontinst replacement for mere mortals and a graphical font installer. These projects are still in an embryonic state.

Taco Hoekwater

CONTEXT System Documentation

abstract
A new website exists that contains documentation for the lower–level ConTeXt macros. The URL for this website is http://tex.aanhet.net/context. This website also contains a full mirror of the pragma-ade website.

keywords
macro programming, context, api, general, documentation

CONTEXT system documentation

All large macro packages for TEX have the need for a number of low–level macros to easy the programming effort. This is definitely true for the CONTEXT package, where the extensive use of key–value pairs and the multilingual interface introduce extra complications to the already tricky art of TEX programming.

There is a large set of manuals and other documentation already available for the user interface of CONTEXT, but nearly no documentation for the TEX–programmer.

Some of the internal macros are real gems, and nearly all of them can also be used in 'normal' source documents. Although most of CONTEXT is described in the source code, sometimes the explanations are too technical for a casual user, and some of the documentation, especially the examples, is still in dutch.

The first series of articles will try to highlight the most general commands of the internal macro layer, using english examples, and removing most of the technical stuff (like the definitions of the macros themselves, and the optimization history).

Later documents will shift focus and start concentrating on questions like: 'How to write a command that uses `\framed` internally' and 'How to add new labels to the multi–lingual interface'.

All of these articles will be available on-line. Using the internet to make this information accessible makes more sense than a document in print. I consider it very likely that the documentation will grow and change over time to insert new examples and document newer macros.

CONTEXT mirror

While I was working on the first article in this series and setting up the needed internet server, the CONTEXT site at http://www.pragma-ade.com was nearly destroyed by hackers. This meant that the main CONTEXT pages were unreachable for quite some time. In order to prevent (or at least) lessen the chance of similar events happening in the future, Hans and I decided to add a CONTEXT mirror/backup site on the new webserver as well.

Where to find it

The website can be reached through the following URL: http://tex.aanhet.net/context/

Maarten Wisse

xml

Hacking TEX4ht for XML Output
The Road towards a TeX to Word Convertor

abstract
This article explains how the author employs the TEX4ht convertor to manage multiple format (XML and PDF) output from a single LATEX source by writing a TEX4ht configuration file and a LATEX class file. Furthermore, it is explained how TEX4ht and the new OpenOffice package can be used to create a new LATEX to MS Word convertor.

Introduction

XML is a hot item in today's IT world. In the area of TeX typesetting, most attention is paid to processing XML input by the TeX typesetter. This is no surprise, because on the negative side, XML input capable typesetters are rare, and on the positive side, TEX does an excellent job in typesetting on demand. Among the mainstream command sets for TEX, ConTEXt seems to have most extensive support for XML output by its recent addition of the XML tag mapping mechanism to the publicrelease.[1]

However, in this contribution, I would like to show that TEX, and more specifically LATEX, can do an excellent job in generating XML output as well. Although originally built as a TEX to HTML converter, the TEX extension TEX4ht can be tweaked in such a way that it outputs arbitrary document type definitions. At this moment, extensive support is contained in the standard distribution for XHTML and MathML, and limited support for the Docbook and TEI DTDs. It is well known that installing and configuring TEX4ht is by no means an easy task. The programming interface is not always intuitive, documentation fragmentary and debugging information rather minimal. Hence, hacking on it is not for the fainthearted, but it must be added that the author of the system is very generous in providing information, solving problems and contributing code in case one does not find a way out of the labyrint.

The Story: What I Wanted and Why

In this article, I want to present two cases in which hacking TEX4ht was valuable for me. As an editor of *Ars Disputandi*, an academic ejournal for philosophy of religion,[2] I developed a configuration file for generating XML output according to the DTD used by the Roquade Publishing project, a joined project of the academic libraries of Utrecht University and Delft University of technology.[3] This was the main reason why I took the effort to learn hacking TEX4ht. Once I managed that problem, I saw a way out of a problem which had bothered me for a long time already: TEX to Word conversion.[4]

Notwithstanding the exciting experience which my use of TEX has been for almost three years, the lack of a good converter – supporting the very special BibTEX configuration I use – from TEX to Word gave rise to numerous problems each time when I needed to

[1] See Berend de Boer, 'From Database to Presentation via XML, XSLT and ConTEXt', in: Simon Pepping (editor), *TEX and META: The Good, the Bad and the Ugly*, EuroTEX proceedings 2001, 27–39.

[2] `http://www.arsdisputandi.org`.

[3] See `http://www.roquade.nl`.

[4] All three configuration files (roqart.cls, roqart.4ht, and ooffice.4ht), can be obtained from my website: `http://www.pmwisse.myweb.nl`.

more or less officially publish research articles in today's WWW: MSWord Wide World. Someone who read an article written by me and published on the basis of a very provisional TEX to Word conversion recently told me that the negation sign in one of the crucial definitions in that article is missing, apparently caused by the conversion.[5] That's a pity—obviously. Furthermore, my provisional way of converting involved manual regeneration of all footnotes – many, in our area of research – in the resulting Word version, something which is time consuming and prone to errors. The phenomenon of 'footnotes' will return in this article.

The story about the Roquade project is somewhat different. The aim of the project is to bypass the commercial publication of academic knowledge by giving the publishing process back in the hands of the university. The project does this by providing a technical XML based epublishing infrastructure which enables scientists to initiate fresh epublishing projects, mainly journals. The long term aim of the project is that the XML generation process is simplified in such a manner that the academic staff could handle it with a minimal amount of training. The current situation is that most XML generation is done by people of the project and this is currently done by linking template based styles in MS Word to XML tags, a conversion which is carried out by a freely available tool Majix. A web based publishing management tool provides editors with XML uploading facilities, followed by a XSLT based HTML output.

As a known techie among the staff of the project, it was suggested that I should try to generate the XML source myself instead of the library staff, much in line with the eventual aim of the project. This is in fact a major advantage for me as an editor, because it provides me with full control over the publication process. I initially started using the Word-Majix procedure, but this quickly turned out to be problematic. First, the procedure lacked necessary robustness, primarily in connection with – yes, again – footnotes. The connection between the styles in the template on the one hand and the XML tags on the other was easily broken by, for instance, a foreign origin (notably WordPerfect) of the Word file. This took me some afternoons to get the XML out of Word. Moreover, the desire to have full and rapid control over the publishing process was hindered by the fact that for every Word to XML conversion, I needed access to a MS Windows machine, because all of my other activities are Linux based.[6] Finally, the desire to have PDF versions of all articles brought TEX into focus as a tool which might generate XML as well as PDF from the same source.[7]

The Roquade Case

I'm not going to repeat all details of the configuration process. The basic steps of tweaking TEX4ht output, along with a special appendix which explains the setup of a new DTD from scratch, can be found in the *The LaTeX Web Companion*.[8] I will only mention those steps which differ from the *Web Companion*. In this section, I will deal with the high level issues, whereas in the final section, I will explain some low level clues which might help people out when developing a new configuration.

The basic thing as explained in the appendix of the Companion is that one creates a .cfg file which fills those hooks needed for a particular setup.[9] I decided to take a twofold

[5] Maarten Wisse, 'The Authority of the Bible', *Religious Studies* 36 (2000), 479.

[6] When it comes to hardware requirements, an additional advantage of my current TEX based setup is that I'm able to edit everything on my recently purchased Intel 386SX Toshiba T2200SX notebook :-)

[7] Yes, I know that the best solution to this problem is to generate both HTML and PDF from the XML source, but due to a lack of human resources among the technical staff of the Roquade Project, this is not to be expected in the near future.

[8] Michel Goossens, Sebastian Rahtz et al., *The* LATEX *Web Companion* (Reading Mass.: Addison Wesley, 1999, 164–184, 404–415.

[9] Goossens, Rahtz, et al., *Web Companion*, 404–408.

approach to my problem. On the one hand, I developed a Latex class file roqart.cls which should take care of the markup of the PDF version of the file. This class file loads article.cls, adding some commands such as `\journalvolume` and `\journalyear`. Furthermore, it automatically generates the articleheading by the `\makehead` command, similar to `\maketitle`.[10] On the other hand, I made a file roqart.cfg which contained all the TeX4ht configuration hooks. Thus, I kept XML and PDF configuration completely separate, as well as completely hidden from the user.[11]

Let me explain my approach by example. The preamble of a Roquade article might look like this:

```
\documentclass{roqart}
\journalvolume{2}
\journalyear{2002}
\title{The Title}
\subject{The Subtitle}
\author{Firstname}{Lastname}
\authorsemail{my@email.com}
\affil{Famous University, UK}
\begin{document}\makehead
```

The remainder of the file is standard Latex. The definition of the `\makehead` command shows how the class file takes care of the different output formats. By an if statement which checks whether pdfoutput is true or false, the class file decides what kind of `\makehead` command to expand. In case of PDF output, the `\makehead` command looks as follows:

```
\newcommand{\makehead}{%
  \vspace*{-1.8cm}\noindent\hspace*{-3.5cm}
  \parbox[b][36pt][t]{5cm}{
    \small\raggedleft\journalname\\Volume \@jrnvol~\@jrnyear\\\issn
    }
  \hspace*{6pt}
  \parbox[b][36pt][b]{6cm}{
    \@logo
    }\par
  \vspace*{36pt}
  \noindent\hspace*{-2.5cm}
  \parbox[t][\height][t]{4cm}{
    \normalsize\raggedleft\itshape\firstname\ \lastname\\
    \footnotesize\raggedleft\scshape\MakeLowercase{\@affil}
    }
  \hspace*{6pt}
  \parbox[t][\height][t]{13cm}{
    \LARGE\raggedright\@title\\
    \vspace*{12pt}
    \ifx\@subject\@undefined\else
    \large\@subject\\
    \fi
    \ifx\@intro\@undefined\else
    \vspace*{24pt}
    \normalsize\@intro
    \vspace{24pt}
    \fi
    }

  \thispagestyle{firstpage}
  \setcounter{footnote}{0}
  }
```

[10] The reason that I did not use `\maketitle` is that TeX4ht uses many special configurations for that command, which make it very difficult to modify.

[11] All package loading, header definition etc. is carried out by the class file as well.

For PDF output, the command generates specific markup. It even automatically adds the logo of the journal on top of the page. When normal LaTeX – the basis of the TeX4ht run – is generated, the `\makehead` command looks like this:

```
\newcommand{\makehead}{%
\PreHead\par\PreTitle \@title \PostTitle\par
\ifx\@subject\@undefined\else
\PreSubject \@subject \PostSubject\par
\fi
\ifx\@intro\@undefined\else
\PreIntro \@intro \PostIntro\par
\fi
\PreFirstname \PreEmail\firstname\ \PostFirstname \PreLastname \lastname\PostEmail
\PostLastname\par
\PreAffil \@affil \PostAffil\par
\PreKeywords \@keywords \PostKeywords\par
\PostHead\par
}
```

The ‘Pre’ and ‘Post’ commands shown in this definition are actually defined as empty elsewhere in the class file. Hence, a normal LaTeX run on them will result in a heading with simple lines of text without special markup. However, when run through TeX4ht, these commands will be redefined to XML tags in the file roqart.cfg, resulting in the following XML output of the `\makehead` command:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE article PUBLIC "" "article.dtd" [
<!ENTITY tab ' '>
]>
<article>
   <h1><ht>The Title</ht>
   <subject>The Subtitle</subject>
   <authors><link url="mailto:myname@email.com">Firstname Lastname</link></authors>
   <biography>Fancy University</biography>
```

Given a valid LaTeX file, generating PDF and XML files of a Roquade article is carried out by typing the following on the command line:

```
maarten@FT190 > pdflatex filename
maarten@FT190 > roqlatex filename 'roqart,html'
```

The latter command executes a shell script which invokes TeX4ht with the appropriate Unicode fonts. This is basically the way it works. It turned out that the procedure described in the Companion, using a .cfg file with the configuration, didn’t work for my setup because it leaves certain internal TeX4ht constructs undefined which are needed by my configuration. Therefore, Eitan Gurari offered me a different way of adding new configurations by making native .4ht files. These are loaded *after* the initial configuration of TeX4ht, and therefore, all basic constructs are available for redefinition.[12]

The TeX to Word Case

Because my TeX to Word convertor was intended to handle the source code of my PhD dissertation files, it had to obey the rules of the standard article class. No new class file was needed. The problems of this case appeared in another area. Let me first describe how I managed the TeX to Word case, because in fact, my convertor is not a real Word convertor, but rather a TeX to OpenOffice convertor. OpenOffice is the name of the open

[12] More information about making .4ht files and the requirements which need to be met in order to function properly can be found in the file mktex4ht.4ht in the TeX4ht distribution.

source project behind Sun's recently acquired StarOffice suite.[13] As soon as I decided to learn XML hacking on TeX4ht, I realized that this was basically important for my conversion problem because of two reasons:

- StarOffice 6.0/ OpenOffice 1.0 would get XML based file formats.
- SO/OO is very good at MS Word conversion.

Hence, if I could succeed in letting TeX4ht output OpenOffice XML, I would basically have Word output as well. The StarOffice 6.0 beta was of excellent quality when it came to reliability and conversion to Word, so that the experiment was worth the job.

The idea was simple. I created a Open Office document to which I added all the markup features which I normally use in my TeX documents, i.e. footnotes (sic!), italics, bulleted and enumerated lists, section, and subsection headings. For the sake of completeness, I also added small caps, boldface, and superscript. I saved that file and began to investigate it in order to sort out the format of the XML.

Now, the most difficult problem emerged, because StarOffice/ OpenOffice does not output a plain XML file – which my TeX4ht convertor does – but, as I figured out later on, a zipped file of a particular structure. At this stage, the open source character of the OpenOffice program became important. On the Open Office site, I found, first of all, a description of the file format, and secondly, an in depth reference manual of the XML DTD at the basis of OpenOffice/StarOffice. I found out that the XML generated by StarOffice is split up into several files, among which are content.xml, styles.xml, and meta.xml, which respectively store the content, styles, and metadata of the file.

The OpenOffice.org site told me as well that StarOffice/OpenOffice is not currently able to read plain XML file conforming to their own DTD. It only reads the zipped files. This seemed the end of my experiment, but it was not. The solution I found was basically simple and obvious. I unzipped my OpenOffice file containing all markup I needed to a new directory, wrote a configuration file ooffice.4ht[14] for TeX4ht producing OpenOffice XML output as found in the content.xml file, replaced the existing content.xml file by the result of the TeX4ht compilation, zipped the file, and loaded it into OpenOffice again.
That worked. Of course, some initial errors appeared, but OpenOffice enormously helped me by telling exactly on which line, which character it found an error in the content.xml input, making the debugging process as easy as possible. Furthermore, the seemingly laborious process of replacing the content.xml file and rezipping the directory into the complete OpenOffice file was easily simplified by a script which automates these tasks.

Of course, my convertor is by no means what people might expect from a full featured TeX to Word convertor. At this moment, it does not support images, tables, formulas, columns etc., although the fact that OpenOffice builds upon MathML means that Eitan Gurari could easily integrate TeX4ht's existing MathML support into ooffice.4ht. The same goes for SVG support. The main thing which ooffice.4ht brought me is native footnote conversion. My `\footnote` commands in Latex become 'real' footnotes in Open Office and, subsequently, footnotes in Word as well. Let me show the configuration hook in ooffice.4ht which does that job:

```
\Configure{footnote}
 {\HCode{<text:footnote text:id="ftn}\FNmark\HCode{"><text:footnote-citation>}\FNmark}
 {\HCode{</text:footnote-citation><text:footnote-body>
   <text:p text:style-name="Footnote">}}
 {\HCode{</text:p></text:footnote-body></text:footnote>}}
```

[13] The OpenOffice Project can be found at http://www.openoffice.org. Initially I worked with the StarOffice 6.0 beta, but because the StarOffice suite will be released under a commercial license, I switched to the OpenOffice 641C (by now 1.0) version. As far as I can see, the solution will contiue to work with both as long as they keep the same DTD.

[14] Ooffice.4ht is actually an elaborated variant of roqart.4ht.

The first and the second argument are put before the footnote text, and the third argument will be put after it. The `\FNmark` command you see in the first argument is an internal TEX4ht command which contains the current footnote mark. It is used twice because OpenOfice adds a `text:id` attribute to the opening tag of the footnote which receives the number of the footnote minus 1. Simply the number of the footnote worked as well. Apart from the footnote support, TEX4ht's reliance upon a real TEX run ensures that all of my special Jurabib formatting features are supported.[15] Finally, TEX4ht and Open Office's extensive Unicode support – although initially somewhat buggy in TEX4ht – means that even Babel based Greek text is properly converted into StarOffice Greek text. The same goes for Hebrew.

Some Clues

So far, I hardly showed any actual TEX4ht configuration code when explaining my XML setups. The reason for this is that it would take huge amounts of space to explain the details of TEX4ht configuration. Much of basic TEX4ht configuration is explained in the LATEX *Web Companion*, and many of the configuration hooks in roqart.4ht and ooffice.4ht are rather simple applications of the information in that book. However, due to the fact that some aspects of configuring TEX4ht are pretty counter-intuitive, I think that it could help users out of a difficult setup process when I explain some tricky issues which are not in the *Web Companion* and nevertheless useful.

The basic idea behind TEX4ht is that it redefines standard LATEX commands in such a way that they receive pre- and post hooks which the user can modify by the `\Configure` command. For example, the standard LATEX command `\textit` is redefined in such a way by TEX4ht that it can be configured as follows (excerpt from roqart.4ht):

```
\Configure{textit}{\Tg<i>}{\Tg</i>}
```

This means that when processed by TEX4ht, the output is not presented in italics (whatever that may be in ASCII!), but preceded by a `<i>` tag and followed by a `</i>` tag, which makes up italics in the Roquade DTD.

This is of course a very simple example. A much more difficult example is the 'paragraph' tag, i.e. the tag preceded and followed by each normal paragraph seperated by the typical TEX-ic blank line. The paragraph hook is a command requiring four arguments, respectively representing the tag before a normal paragraph and an indented paragraph, and the tag after a normal and an indented paragraph. Hence, taking an example from the ooffice.4ht configuration file, one finds the following definition:

```
\Configure{HtmlPar}
   {\EndP\HCode{<text:p text:style-name="Text body">}}
   {\EndP\HCode{<text:p text:style-name="Text body">}}
   {\HCode{</text:p>\Hnewline}}     {\HCode{</text:p>\Hnewline}}
```

The first two arguments contain the strange code `\EndP` appears, a code which is omnipresent in TEX4ht files and typically counter intuitive. Given the name 'EndP' one would expect that it appears in the last two arguments of the paragraph hook, but it does appear at the start of the first two. The *Web Companion* explains that 'The task of `\EndP` is typically to deliver code from the start of a paragraph to its end.',[16] but this does not help us much further.

In some way or another – Eitan had his reasons for it, of course[17] – the `\EndP` command works from the start of a new paragraph in order to terminate the *previous* paragraph and place its closing tag (e.g. `</text:p>`). Hence, the `\EndP` is placed in the arguments of the

[15] For more information about Jurabib, see `http://www.jurabib.org`.

[16] *Web Companion*, 183.

[17] Eitan: 'The reason is that TEX offers access to the start of paragraphs (through `\everypar`) and not to their ends :-('

pre tags in order to place the post tag of the previous paragraph when a new paragraph is started. A similar trick is used for the `\IgnorePar` command, which is frequently found before or after `\EndP` commands in TeX4ht configuration files. This command, used in conjunction with `\EndP`, lets the previous paragraph end and starts a new one, but dumps the pre and post code of that paragraph, which would normally appear. Let me finally give two code examples which illustrate this trick:

```
\renewcommand{\PreSubject}{\EndP\IgnorePar\HCode{<subject>}}
\renewcommand{\PreIntro}{\EndP\IgnorePar\HCode{<intro>}}
\renewcommand{\PreKeywords}{\EndP\IgnorePar\HCode{<keywords>}}
\renewcommand{\PostSubject}{\HCode{</subject>}}
\renewcommand{\PostIntro}{\HCode{</intro>}}
\renewcommand{\PostKeywords}{\HCode{</keywords>}}
```

These are the redefnitions which fill up the empty commands defined in roqart.cls with the appropriate XML code for the Roquade DTD. When we look at the code in roqart.cls, we see that every line is terminated by a `\par` command. In the code from roqart.4ht, we see that every pre command reckognises this `\par` command by executing a `\EndP` command. However, given the fact that the default paragraph hook has been configured to start each paragraph with a `<p>` and corresponding `</p>` tag, this would mean that the tags on top of the article would get it as well, which is not correct in this DTD. Therefore, each pre command gets a `\IgnorePar` command as well which drops the `<p>` and `</p>` code.

Finally, let's go into a really difficult problem from a TeX4ht configuration perspective, i.e. configuring a simple itemized or enumerated list. We take the configuration of the itemize environment from roqart.4ht as an example. The XML code of an itemized list according to the Roquade DTD is kept very simple. It looks like this:

```
<list type="itemized">
<item>This is an item</item>
<item>This is another one</item>
</list>
```

The TeX4ht code which produces this XML code looks as follows:

```
\ConfigureList{itemize}%
   {\EndP\HCode{<list style="bullet">\Hnewline}\def\end@Item{}}
   {\EndP\HCode{</item></list>}\ShowPar}
   {\EndP\end@Item\DeleteMark}
   {\HCode{<item>}\par\ShowPar \def\end@Item{\Tg</item>}}
```

The *Companion* describes the `\ConfigureList` command as:

```
\ConfigureList{name}{pre-list}{post-list}{pre-label}{post-label}
```

The different lists of Latex carry the hooks in the following manner:

```
<pre-list hook>
<pre-label hook> MARK <post-label hook> CONTENT OF ITEM 1
<pre-label hook> MARK <post-label hook> CONTENT OF ITEM 2
...........
<pre-label hook> MARK <post-label hook> CONTENT OF LAST ITEM
<post-list hook>
```

To allow marking at the end points of the content of the items (e.g., `</item>`), we must attach those marks to the `<pre-label hook>` and `<post-list hook>`. However, the first `<pre-label hook>` should not carry an end mark. Moreover, in the boundary case of empty lists with no specified items, the `<post-list hook>` should also avoid producing such an end mark. The `\end@Item` helps handling these cases correctly, and the `\EndP` takes care of in-time providing the closing paragraphs for those opened within the items.

In some cases, like in itemized and enumerated lists, but unlike in the cases of description lists, we don't want the marks provided by LATEX. Instead, we expect them to be created by the interprets of the XML code. The `\DeleteMark` comes handy here.

The LATEX model of paragraphs is not always in line with the model of paragraphs in the XML standard in use. The `\IgnorePar` and `\ShowPar` have been introduced to help bridge the differences. The `\par\ShowPar` forces a start of paragraphs at the start of each item, allowing to introduce the `<p>` tag there. (The LATEX model allows paragraph breaks withing the content of items, but not at their start points.)[18]

Conclusion

It might have become clear to the reader that the configuration of TEX4ht is no easy task. However, I hope that it is clear as well how extremely useful the system can be once properly configured. Equipped with this convertor, I manage both an e-journal and TEX to OpenOffice/MS Word conversion with little effort.[19]

[18] Thanks to Eitan for offering this explanation of the `\ConfigureList` problem.

[19] I would like to thank Eitan Gurari for his contribution to and comments on this article.

Hans Hagen
Karel H Wesseling

The authors are indebted to Wybo Dekker for his most valuable suggestions, for proofreading and for numerous acts of additional help.

context

TEXEXEC User's Guide

abstract
This guide describes the uses and options of the texexec program that is available in the Context distribution. The options are invoked by calls on a command line, which are words preceded by two hyphens as in `--make`. There are options for running Context on your TeXfile to produce printable output, options to specify languages, an option to make listings of (software program) files word for word, options for conditional execution, for selecting pages to print, for printing on differently sized paper, for directing your output to a particular file, for conversion of SGML and XML to TeX. If it is no problem to you to use a command line and to occasionally look things up in the help file or in this user's guide, you will find texexec to be a useful, even indispensable tool for Context.

Samenvatting
Dit is een gids voor de gebruiker van het "texexec" programma dat met Context samen gebruikt wordt voor een hele serie handigheidjes en acties, en bijna onmisbaar is. Texexec leest een commando regel waarop woorden staan voorafgegaan door twee mintekens zoals in `--make`. Zo'n commando heet optie en er zijn opties voor het compileren van een Context brontekst, opties voor het specificeren van talen, voor het maken van listings, voor voorwaardelijke compilatie, voor het selectief compileren van pagina's, voor het printen op andere formaten papier of scherm, voor het in vreemde volgordes plaatsen van pagina's zoals dat nodig is voor het vouwen van boekjes uit grote vellen. U mag niet schrikken van commandoregels en moet soms dingen opzoeken in een help bestand of in dit stuk. Maar dan heeft U ook een gereedschapskist van belang in handen.

Kick start texexec

It takes until page 40 before we actually reach the parts where your TeX file will be processed. For the impatient we provide the ultimate quick start here, under the assumption that you are using the English language, which is, after all the "lingua franca" of modern science, and under the further assumption that your TeX file name is `lingua.tex`.

Prepare the necessary files for TeX and for Context to work with the English language:

```
texexec --make en
```

Process your file with Context to produce a PDF file for viewing with Acrobat or Ghostview:

```
texexec --pdf lingua
```

What is texexec used for?

Texexec is a so-called "command line" interface for Context, directing it. Context is a typesetting computer system that extends TeX with easy to understand operations activated by typing an `\instruction[with][parameters]{and text}` such as `\inframed[height=fit,width=fit]{which looks boxed}` which looks boxed. In the next sections you will see more examples of such instructions. The instructions are programmed in TeX by the author of Context and form — as it were — a higher level "shell" around TeX. With Context you can do very complex things with relative ease.

To carry out the instructions placed between the normal text in your TeX file Context processes that file to produce typeset output as a DVI or a PDF file. Through texexec

can it be controlled to do the right thing by supplying it with commands.[1] In addition, a series of very useful functions or 'options'[2] are also built–in, as you will see later, and in directing the processing of your TEX file it also shows substantial intelligence. Texexec has become a veritable workbench with so many options that a user's guide is needed.
But you need access to a command line. Under Windows that means back to DOS. Move your mouse pointer to the task bar and click "Start", move up to "Programs" and follow the arrow, in the next window go up to "MS–DOS Prompt" and double click. A dark window opens showing a DOS prompt where you can type your command lines and have them executed. To close the window type `exit`. A nicer solution is to install the shareware program Windows Commander. This system gives you a command line with a simultaneous overview of your files and offers many features. Download it from `www.ghisler.com`. Installation is almost automatic.

Texexec started out as a stand alone program that worked as DOS software. In later years demand for the program to work on other platforms such as Linux and Mac and Windows rose and it was decided to rely on a language to develop texexec, that is almost universally available: perl. Thus, you must have perl installed on your computer. If you don't have perl go to the perl site to obtain it. Installation is simple and reliable.[3]
Once perl is there you can run texexec and direct CONTEXT and enjoy its power and its many features.

Many languages, fewer interfaces

When you are a TEX user you were perhaps attracted to it because it supports so many languages with near perfect hyphenation. In this guide and texexec we call a text language "language" as you'd expect. There is also an instruction language. CONTEXT, when processing your TEX file for typesetting, searches for type setting instructions embedded in your text. They are almost always preceded by a backslash, \, and easily recognized. Instructions are words or short phrases typed in English, but CONTEXT can be told to work in one of several instruction languages, called "interface". For example in English you'd start a new chapter in your text with the command:

```
\chapter{Installing perl}
```

However, if you were Dutch but writing an English user's guide, and CONTEXT was prepared to listen to Dutch commands you might have typed:

```
\hoofdstuk{Installing perl}
```

where the Dutch 'hoofdstuk' and the English 'chapter' have the same meaning and invoke the same actions. CONTEXT, when just installed, thus has to be taught an instruction language or interface such as Czech, Dutch, or English.

To teach CONTEXT more than one interface and generate the appropriate configuration files execute the command:

```
texexec --make nl en de
```

The `make` command is the first command you use with texexec and before its execution CONTEXT will not work. Note that we are talking about the CONTEXT interfaces and not about the text language. The corresponding language's so-called "format files" for TEX

[1] The word 'instruction' is used throughout when we refer to an instruction to CONTEXT proper, 'command' is used for instructions to texexec as given on the command line.
[2] An option is a word preceded by -- that causes texexec to change from default processing to optional processing.
[3] Installation of TEX, CONTEXT, texexec, or perl is not treated in this document.

to use are automatically generated with the interfaces. Available interfaces at present are coded as:

`nl`–Dutch,
`en`–English,
`de`–German,
`cz`–Czech,
`ro`–Romanian,
`it`–Italian.

CONTEXT is multilingual with more than 20 languages supported. This topic is treated in the CONTEXT manual. Clearly, there are more languages than interfaces. It is possible to prepare your system for several more languages simultaneously with a limited number of interfaces, and add special fonts at the same time. For example, some Slavic text languages require special fonts. With english as the instruction language you type:

```
texexec --make --language=pl,cz,sk --bodyfont=plr en de
```

which gives you two interfaces (English and German) and five languages (English, German, Polish, Czech, and Slovak), with `pl,cz,sk` signifying Polish, Czech and Slovak languages; `plr` is the Polish alternative for cmr, Knuth's Computer Modern Roman font; `csr` is the cmr for Czech & Slovak languages. When one or more additional languages are specified then the first after the `language=` sign becomes automatically the default language for hyphenation and for TEX. In the above example, thus, Polish.

`--interface=languagecode`
Your TEX file contains pure text, mixed with the necessary typeset instructions to CONTEXT. Do not mix CONTEXT instructions of different languages. Texexec, normally, can find out which interface language you used, but it can also be told which one it is with the command `--interface`. In that case CONTEXT will only react to instructions in that particular language and report errors for unrecognized ones. Specify this on the texexec command line as:

```
texexec --interface=de test
```

The accepted language codes are mentioned in section 3 on page 37. It is more appropriate, however, to specify the CONTEXT interface language in the first line of your TEX file with:

```
% interface=en
```

To establish an interface language texexec first checks your command line, then the first comment line of your text (which takes priority), then — but only if no interface language was found — it checks the language of the first several instructions (such as `\starttext`) in a final effort to establish the interface. If still unsuccsesful the english interface is assumed. To avoid any error or ambiguity: just specify it.

`--response=languagecode`
CONTEXT not only understands several interface languages but can respond with messages in the same languages but only after it has been prepared for that. Use the `--make` command and add on the command line:

```
texexec --make --response=de
```

In your texexec and CONTEXT runs from then on you get a log file with messages in German, although the messages of TEX itself are still in English.

Even though with the `--make` command there has been a default language established it is better to specify the text language of your document with a CONTEXT instruction, if only to ascertain a proper hyphenation. To set English as the principal language for your text you would use `\language[en]` somewhere before the start of your text. To have the hyphenation rules and exceptions available for the language of your specification, they must have been "made" with the `--make` command. Since CONTEXT is multilingual you can switch languages within the document. The easiest way is to isolate with curly braces the section with the deviant language and specify the local language in shorthand notation as follows: `{\de Die Schwefelmeiler f\"{u}llten Berg und Tal mit giftigen D\"{u}nsten.}` A clearer way for longer sections of deviant language text to switch to another language is offered by the `\start ... \stop` mechanism of CONTEXT:

```
\start \nl
In Siddeburen was een bok die machtsverhief en worteltrok. Die
bok heeft onlangs onverschrokken de wortel uit zichzelf
getrokken, waarna hij zonder ongerief zich weer in het kwadraat
verhief. Maar 't feit waardoor hij voort zal leven, is dat hij
achteraf nog even, de massa die hem huldigde met vijf
vermenigvuldigde. Een "Trijntje Fop".
\par      % or add an empty line before \stop as below:

\stop %nl
```

`--alone`

With this option texexec must do its work on its own and not call external functions or utilities such as `fmtutil`. Such callings may — on occasion — lead to errors if such a called system is not prepared for CONTEXT. This is basically an installation and error tracing option. A typical application is:

```
texexec --make --alone metafun
```

ASCII and other text encodings

In central Europe, in countries as Poland or the Czech Republic many special characters are used perhaps with so-called diacritical marks. For example a Czech name is Peňáz which may be typed as `Pe\v{n}\'{a}z`. If most of the words you type are like this TEX quickly becomes a nuisance: the possibility for special characters is there but the facility is difficult to use, a contradiction in terms.

Originally TEX used 7–bit ASCII. ASCII is the American Standard Code for Information Interchange. It needs only 7 bit for each of the 128 characters that it supports. But modern computers are byte oriented and a byte has 8 bits. This leaves one bit and thus halve of the possible number of characters per byte unused. Thus so-called 'code-pages' have been defined and each code page has an identifying number and special meanings for the other 128 positions in each byte. Switching code-pages is not a very nice process and it may lead to silly mistakes and strange characters printed on screen or on paper. A more recent solution has been to design a Unicode which uses two byte per character and thereby has possibilities for 65536 characters, special symbols and glyphs.

CONTEXT supports the more commonly used encodings as specified in files in the CONTEXT directory tree beginning with `enco-`, `lang-`, or `font-`. If you are interested please print these files. In recent TEX versions generically called WEB2C it is possible to specify a conversion or translation called mapping from the non–ASCII document encoding that you might use to a standard encoding. This mapping also takes care of fonts and hyphenation patterns. Use texexec to command such a mapping on its command line but preferably in the first line of your file as it is then permanently documented as non–standard:

```
%  translate=cp1250pl
```

or

```
texexec --translate=cp1250pl nontest
```

`nontest` is a file, unlike our test file (see chapter 5), which has a non–ASCII encoding.

CONTEXT and your test file

Assume that you have prepared a file with an ASCII oriented editor which contains a text and some typesetting instructions in CONTEXT. This file is called `test.tex`. A short but complete one is shown below:

```
% output=pdftex
\setupoutput[pdftex]
\starttext
\title{A very simple test file for \CONTEXT}

This file is used initially as a test file to verify that
\type{texexec}, perl, and \CONTEXT\ work as expected. We could
easily add more words but there is no need.

\stoptext
```

Now type on a command line:

```
texexec --pdf test
```

This starts texexec, tells CONTEXT to work on `test.tex`, and to produce PDF output, readable and printable with the Acrobat viewer, and usually also Ghostview. Texexec first checks its command line (one command only in this case, except the input file) and next reads the first line of your test file. If it begins with a `%` symbol it is read and checked for commands that it might understand. In this case there is a command that the PDF output format must be generated, again. By default, however, a DVI output format is generated. DVI is a proprietary TEX format that is device independent and can be viewed and printed with DVI viewers though not with Acrobat. The Linux operating system often comes with a DVI viewer already present. There are two other ways of overriding the DVI default and they can be used all at the same time although just one form suffices.

The texexec command line option for PDF output is:

```
texexec --output=pdftex test
```

or shorter:

```
texexec --pdf test
```

In your TEX file you might also type a line:

```
\setupoutput[pdftex]
```

which was already included in the test file as shown above. We now know that one of the solutions would have been sufficient but all three together does no harm.

Importantly, the last output specification that is found rules the form. Thus, if you have `--pdf` on the command line but `\setupoutput[dvips]` in the setup area of your document the output will be DVI.

Keeping CONTEXT under control

Intelligence slows down

As mentioned, texexec has a certain intelligence built–in. For example, assume that you work on a very substantial text which includes a table of contents and an index of terms. These items require a certain space, perhaps even a number of pages. But beforehand the exact space required is not known. After your TEX file is processed once, the contents and number of index terms is known. In a second run these tables can be typeset. Then, finally, in a third run all page numbers are known and can be entered in both tables. Texexec observes what is going on during the typesetting and has ways to determine how many CONTEXT runs are needed.

But assume that each run takes 1 minute, and assume that you are impatient, and further assume that you are only interested in checking that you have entered your CONTEXT typesetting instructions correctly. In that case a single run suffices. Type a command line as follows:

```
texexec --once test
```

or

```
texexec --runs=1 test
```

The latter command suggests that `--runs=2` is also an option, which it is.

Finally fast

Some fast thinkers complain that CONTEXT takes its time when processing. If considering how much you get in return does not give sufficient relief of the symptoms then use the `--fast` option. Certain typesetting actions are quite time consuming. The inclusion of graphics files as illustrations is notorious, but the output file without the graphics does not look nice. But you are impatient and willing to pay the price. Then use this option and such time consuming actions will be skipped. A graphic will be replaced by a box outline in the proper size, so that the overall layout is not affected. Your last run, however, should contain the `--final` command to ensure a complete final result, skipping nothing, and doing an adequate number of runs automatically.

Complexity options

Even more complex: texexec commands CONTEXT and CONTEXT calls TEX. Assume that there is a reason to specify one of several TEX text processor programs. Then specify it on the texexec command line:

```
texexec --program=tex --format=plain test
```

There may be a conflict with a specification for PDF output that plain TEX cannot generate. But the conflict is resolved by texexec, and DVI output is produced instead. An alternative is to use the first line in your TEX file as the command line for texexec (without the `--`). An example of such a pseudo command line is:

```
%  interface=en program=pdfetex output=pdftex
```

or

```
%  interface=en program=pdfetex output=dvips
```

depending on your desired output format.

`--batch`
Batch processing means that CONTEXT or TEX (see page 36) does not halt to ask for user input when encountering an error. It places any errors and questions together with line numbers in the log file while continuing the processing.

`--convert=fileformat`
CONTEXT expects its own typesetting instructions in the text file. But TEX and CONTEXT are not the only typesetting systems. Other world wide standards are SGML and XML. With use of texexec XML and SGML mark up can be translated to genuine CONTEXT and TEX. For example you may place XML instructions in your text and then use the `--convert` option to translate:

```
texexec --convert=xml testxml
```

`--format=formatfile`
When compiling, texexec uses the CONTEXT format files (files beginning with `cont-`). It is possible to use other format files, when testing. This is a rather specialized option that you might not use often. An example is:

```
texexec --format=plain --program=pdftex somefile
```

Fabulous flexibility

`--color`
This turns on typesetting in color. The alternative is to use the CONTEXT instruction `\setupcolor[state=start]` in the setup area of your TEX file, before the `\starttext` instruction. Such instructions in the TEX file override the command line option since they are processed later. Assume, however, that you have a document in which you have emphasized portions using colors. Assume you have a black ink LaserJet on which a color output file would cause undesirable gray scales from dark to unreadably light perhaps. In that case, don't turn color processing on in your text but in a final run, before going to an outside color printer, produce a result file with the `--color` option. You may also use the `--mode` command on page 45.

`--environment=listofnames`
An environment is usually a separate file with CONTEXT instructions which is placed up front in your TEX file to specify the make up of your text pages. For example, you might make an environment file named `e-scr.tex` which contains the single CONTEXT instruction `\setuppapersize[S6][S6]` which would typeset your document with sizes best suited for viewing on screen. Default, your output would be for A4 paper. Normally you would place reading of the format file in the setup area of your document. This time don't, but put it on the texexec command line, and increase your flexibility in processing:

```
texexec --environment=e-scr test
```

Your output result file will be optimal for screen viewing. But with a different environment file it could be made optimal for A5 size paper, yet your document itself would be identical to the comma. Note: this particular effect can be achieved more easily with the `--mode=screen` command explained on page 45 but the general principle holds, and environment files may attain great complexity.

`--output=driver`
The use of this command is already explained in section 5 on page 40 where is was used to produce PDF output (with the option `--output=pdftex`, or its abreviated form `--pdf`).

Further possibilities are `--output=dvips` which produces the default DVI output and need not be specified, `dvipsone`, `dviview` which is an experimental viewer, and `dviwind`.

`--result=outfile`
This option is used a number of times in this guide, for example in combination with the `--mode` option, see page 45. With it you set the name of the output file to which the result of the processing is written. Addition of the proper filename extension is automatic.

METAPOST measures

METAPOST is a language to draw vector graphics, and its output can be Encapsulated Postscript (`.eps`) which can be incorporated in a CONTEXT document.[4]
METAPOST instructions can be embedded in a CONTEXT, and with a little more trouble in a LATEX document. These METAPOST instructions do not start with a backslash like TEX's and would be typeset as text instead of compiled as METAPOST graphics. So, therefore, in a CONTEXT document you must surround them with instructions such as:

```
\startuseMPgraphic
...
any number of MetaPost instructions
...
\stopuseMPgraphic
```

When CONTEXT reads this file and meets the METAPOST code it writes the code to a file. There are now two possibilities or options, the direct option and the indirect option. In the direct option CONTEXT calls METAPOST, has the graphic made, and upon return places the graphic in the document. This process is repeated for any METAPOST graphic it discovers. Thus, the option is relatively slow, yet it is also the default option. In the indirect option, the METAPOST is similarly extracted but not compiled until after the CONTEXT run finishes. Texexec detects that METAPOST files have been generated and calls METAPOST to produce the graphics which are placed in yet another file. (This processing can be blocked with the texexec command `--nomp`.) When not blocked texexec calls CONTEXT a second time when the prepared graphics is included. The indirect option is not default, but it can be forced with the texexec command `--automp`. Thus, when you have embedded METAPOST in your document with many single graphics use the fast indirect option as follows:

```
texexec --pdf --automp --result=oneforal metatest
```

Passing the word

`--passon=string`
Not only texexec uses options but so do some TEX implementations. MIKTEX, for example, uses the option `--src` so that editors can set their cursor position to a location in the typeset DVI file. You may pass this option to MIKTEX as follows:

```
texexec --passon="--src" miktest
```

The quotation marks are mandatory.

[4] See: Fabrice Popineau: Practical METAPOST in this issue.

Optionally list

`--help`

This option produces a rather terse list of all texexec options with some further comment which can be useful for experienced users or when you remember that a function is available but don't recall the magic word exactly:

```
texexec --help
```

This produces a list of options that on a Windows systems flashes by so fast that there is no time to read. Use DOS commands to get some control over this process. For example, use the DOS redirect symbol > to put the list to a file `help.txt` or use the `| more` command to get it in pages. More help on a particular option can be evoked with:

```
texexec --help pdfselect > helpsel.txt
```

which produces a short list of sub–options for pdfselect and places it in the file `helpsel.txt`. There is no extra help on help.

`--listing`

This useful option produces a typewriter–look output of your file as it is shown in your text editor. You get a DVI file unless `--pdf` is placed *after* the `--listing` option. The file to be listed must be given with its extension, or the action is cancelled. Listing is not limited to CONTEXT files with extension `.tex` and the files do not have to contain any CONTEXT instruction, and when they do they are shown verbatim:

```
texexec --listing --pdf detect.mod
```

produces a file `texexec.pdf`. You may specify your desired output file name with the result option:

```
texexec --listing --pdf --result=testlist test.tex
```

produces the output and its log file as `testlist.pdf` and `testlist.log`. It is not wise to name this listed file `test.pdf` since its normally typeset output would also have that name and overwrite your listed file.

Naturally, you may apply `--listing` to TEX, perl, or METAPOST software program files. By commanding `--pretty` on these files you invoke "pretty" printing, meaning that language items become highlighted in the program text, even highlighted in color if you specify `--color` as well:

```
texexec --listing --pretty --color --pdf softwa.pl
```

with the resultant output in `softwa.pdf`. Finally, `--backspace` and `--topspace` may be used as shown on page 47.

`--module`

CONTEXT as a software program is written in the language TEX designed by Donald E Knuth many years ago, and it was set up in modules. Some of these modules can be found in subdirectories such as, for example:

```
c:\4tex5.0\texmf\tex\context\base\
```

unless your CONTEXT system was not placed in `c:\4tex5.0\` but somewhere else. Now go the `\base\` subdirectory and then type on your command line:

```
texexec --module --mode=color --pdf colo-ini.tex
```

The result of this processing is found in colo-ini.pdf and in color it allows you to view (and print) a CONTEXT module and how it is programmed. You may like to view this one even as a user, non–programmer.

`--silent and -verbose`
Texexec produces a so-called 'log' file in which it places phrases that express what it is doing, problems that it encounters, and errors that may occur. The same information is also shown in a text box on screen but this often passes by so fast that it cannot be understood. The logging of CONTEXT can be normal, which is the default, or it can be switched to verbose thereby producing log files which also contain information possibly loaded from the texexec.ini file, or to silent which produces a smaller log file. This is done as follows:

```
texexec --verbose test
```

or

```
texexec --silent test
```

For the smaller size TEX files the differences between these log files is insignificant.

Those powerful modes

With mode is not meant that which occurs most frequently or which is fashionable but modus operandi, a way to operate, a manner of doing something. Mode gives you power. Please read on.

`--mode=mysize`
Default, CONTEXT prepares your typeset document for printing on A4 paper, but other paper sizes can be accommodated simply with a `\setup...` instruction such as `\setuppapersize[S6][S6]` (S6 is just a code), which prepares your document for display on a computer screen. If `\setuppapersize[][]` is absent in your document you may specify a "paper" size as a texexec command as follows:

```
texexec --mode=screen test
```

which produces the desired screen sized output. But note that a `\setuppapersize[][]` instruction in your TEX file has precedence since it will be read later. Given the `--mode` option it is now simple to produce two or more differently sized output files:

```
texexec --mode=screen --result=test-scr test
texexec --mode=A4     --result=test-a4p test
```

which produces the files `test-scr.dvi` and `test-a4p.dvi`, unless you also added the `--pdf` option in which case you obtain `.pdf` files. Predefined modes are `A4`, `letter`, `legal`, `color` and `screen`. As far as European paper size are concerned, all sizes from A0 down to A9 are options.

Once you have set the mode you can use the `\doifmode{truetest}{true-instructions and/or text}` instruction in your text to accomplish special things, as follows:

```
\doifmode{screen}{\setupcolor[state=start]}
```

which seems useful if you have a color screen display and a black ink printer. The same can be achieved with the `--color` option and with `--mode=color`.

`--mode=mymode`
The `--mode` option, however, is more powerful than this since you can define your own mode by specifying one in your text, as follows:

```
\startmode[myspecial]
  \environment e-special
 \stopmode
```

With:

```
texexec --mode=myspecial test
```

ConTeXt, when examining your text file and seeing a `\startmode` instruction, will compare bracketed names and execute the instructions between `\startmode[]` and `\stopmode` only when the names match. Thus, you now have the option to conditionally have ConTeXt execute instructions to your taste. It is not limited to loading an environment, as shown above, which can be done equally conveniently with the `--environment` option, see page 42, and it is not limited to one single mode but you may specify many modes as a comma separated list:

```
texexec --mode=myspecial,mymode,screen --pdf test
```

Similarly, you may put a conditional ConTeXt instruction in your text as follows:

```
\doifmode{myspecial}{\page}
```

which forces a new page at that point if and only if the mode is `myspecial`. Any series of instructions and text to be printed can thus be made conditional. For an IF there is an ELSE. You may also have in your text:

```
\doifnotmode{myspecial}{\page}
```

Weird arrangement of pages

`--pages=listofpagenumbers`
For double sided printing on printers that do not support this feature automatically the odd numbered pages should be printed first, then the even numbered pages. This can be achieved by making two output files and printing them one after the other with this option as follows:

```
texexec --pdf --pages=odd  --result=odtest test
texexec --pdf --pages=even --result=evtest test
```

This effect, however, can more easily be achieved by using Acrobat as a printer controller.

You may also specify a comma-separated list of page numbers such as `5,2,99,125`. The specification list can be in any order but the resultant output is in increasing order of page number. A range of pages can be given as `2:6`. Finally, both forms can be mixed at will, thus `5,7:17` is also valid.

`--noarrange`
ConTeXt has many possibilities to arrange pages on a sheet of paper. This process is called “page imposition” and often quite complex. It is, for example, possible to arrange A5 pages on A4 sheets in such a way that after printing and folding they form a booklet. The following ConTeXt instructions:[5]

```
\setuppapersize     [A5][A4]
\setuparranging     [2UP,rotated,doublesided]
\setuppagenumbering[alternative=doublesided]
\setuplayout        [margin=0pt,widt=fit]
```

[5] This example is taken from a ConTeXt manual.

```
\starttext
  ....
\stoptext
```

produce such a booklet. Such arranging is something to delay to the final run. For initial runs the arranging can be suppressed with the `--noarrange` option so that the result can be more easily viewed on screen, with pages in normal order. Similar arranging can be achieved by passing texexec the `--pdfarrange` option, see page 47. With the instructions above, however, typesetting is done with specified fontsizes. Thus, if you specified a bodyfont at size 10pt your text would come out rather large for an A5 sized paper. To achieve an effect more similar to `--pdfarrange`, which photographically reduces the size of a page with the type size, you should specify `\setupbodyfont[7pt]`. The command `--pdfarrange` performs no typesetting.

`--arrange`
This option assures that CONTEXT arranges pages on sheets in an order proper for folding to a booklet only in the last pass when everything else has been properly set, such as referenced page numbers. For example, you may have specified in your TEX file that CONTEXT should arrange output as two A5-sized pages per A4-sized sheet printed double sided. For such booklets to fold properly the order of the pages has to change from normally increasing to weird.

Shuffling PDF pages

Arrange with `--pdfarrange`

Once a PDF file has resulted from CONTEXT processing, the pages can be rearranged in that file in such a way that the rearranged file can be printed as a booklet. The next command, for example, rearranges the pages of the well known TEXLive manual. Note that the command works on a `.pdf`, not a `.tex` file, and the extension must be specified:

```
texexec --pdfarrange --paper=a5a4 --print=up live.pdf
```

With `--pdfarrange` and `--paper=a5a4` the pages are reduced in size and the font size with it. This option works as if you did a photographic size reduction. Without `--print=up` the result of the above is A5 sized pages nicely centered on A4 sized paper sheets, one per sheet. With `--print=up` you get two A5 sized pages per sheet ready for double sided printing which Acrobat can do for you. The output can be found in file `texexec.pdf`. The output can be directed to a filename of your choice with the `--result` option. The pdf extension is automatic and does not have to be given.

The `--pdfarrange` option can be combined with a number of other options to fine tune positioning of the pages on the sheets:

```
  --backspace   inner margin on paper sheet
--paperoffset   room left over at paper border
  --textwidth   width of original text page
   --topspace   top and bottom margin on sheet
```

They have to be specified in a dimension known to TEX such as `pt, cm,` or `in` and values can be positive or negative, for example: `--paperoffset=-1cm` is valid. It reduces white spaces at the border which leaves room for a wider and higher printed text page. These options require some experimenting and it is suggested that once a pleasing set is found you save them in a file for future referencing.

But this is not all. With `--addempty` you may add an empty page after a series of specified page numbers in the original numbering scheme. For example `--addempty=3,9,55` is valid and adds an empty page after page 3, page 9 and page 55. The page numbers do not change. These pages may appear on sheets numbered 1 and 17, for example. Single sided printing is commanded with `--noduplex`, and cut marks may be added with `--markings` on the command line.

`--paper=keyname`
This option and the `--print` and `--pdfarrange` option act together to do things similar to what can be achieved as explained under `--noarrange` on page 46. Paper options are `a5a4` and `a4a3`, the former for A5 size pages printed on A4 sheets, the latter for A4 sized pages printed on A3 sheets. Any other European paper size combination can be used from A0 down to A9.

`--print=keyname`
This option and the `--paper` and `--pdfarrange` option act together to do things similar to what can be achieved as explained under `--noarrange` on page 46. For `--print=up` you get two pages per sheet, doublesided, whereas `--print=down` causes two 90 degree rotated pages per sheet, doublesided.

Combine with `--pdfcombine`

This option allows you to print several pages on one sheet. If your CONTEXT setup specified printing on A4 paper and the paper format to print on is also A4, the size of each page is reduced or there would not fit more than one page on a sheet. Specify the arrangement and number of pages on one sheet with `--combination=2*2` for example. The number of columns in a row comes first, the number rows of pages comes second. Although `--combination=1*2` or `2*1` are certainly allowed they leave much empty space on the sheet with room to fit two more pages. The `--print=up` option (see page 48) does not work here. Thus, combinations from `2*2` onwards are useful. With further compression, however, the print quickly becomes unreadable.

The `--paperoffset` may help in keeping combined printed pages readable. The paper offset specifies a rim of white space around the printed sheet. The smaller the rim the more space is available for printing. Thus `--paperoffset=-1cm` compared to the default `+1cm` helps.

Copy with `--pdfcopy`

On photocopiers you can make reduced and also enlarged copies of originals. You only have to make sure that the copy fits the paper size and if necessary adjust the position of your original on the glass plate. Such copying is available with texexec:

```
texexec --pdfcopy --scale=.707 --result=test07 test
```

produces a $\sqrt{0.5}$ sized photocopy in file `test07.pdf` neatly centered on the page, and note that keeping pages parallel on the sheet is automatic unlike with most photocopiers.

```
texexec --pdfcopy --scale=1.414 --result=test14 test
```

enlarges the printed area with, at this scale, sections falling off the paper. The remaining text is upper right adjusted on the sheet.

```
texexec --pdfcopy --scale=1.5 --paperoffset=-10cm test
```

enlarges your page 50% and removes any empty borders but the page is still right adjusted on the sheet and no option is available to preferentially keep another area of the page on the sheet.

Select with `--pdfselect`

This useful option lets you cut one or more pages from a `.pdf` file and collect them to another file. Use it in combination with `--selection=listofpagenumbers`. It is even possible to place the selected pages on a different page size by using the `--paper=keyname` option. Note that the option works directly on any `.pdf` file but not on the `.tex` file which possibly was the source of the `.pdf` file. Here follow four examples of command lines:

```
texexec --pdfselect --selection=1,2,5 test
texexec --pdfselect --paper=S6 --selection=1,2,5 test
texexec --pdfselect --selection=1:5 test
texexec --pdfselect --selection=1,2,5:11,17 --result=subrange test
```

The first command line cuts pages 1, 2 and 5 from the `test.pdf` original and places them (by default) in `texexec.pdf` with the original page numbers retained. The second command line cuts the requested pages from `test.pdf` and places them resized to fit and left adjusted on screen sized "paper" in `texexec.pdf`. You may make reduced size booklets by specifying `--paper=A6`, and enlarged ones by specifying `--paper=A2`, which is probably beyond the possibilities of your printer. The third command line cuts pages from 1 to 5 inclusive. The fourth command line cuts pages in more complex specification to direct its output to `subrange.pdf`.

Thus, whenever in the future, you'd like to extract a table or a figure or the summary of a paper you have written, cut the appropriate page(s) from your electronic PDF type document and have them printed, just like you used to do by selectively photocopying such pages from the printed document.

Option

Index

Wybo Dekker
wybo@servalys.nl

latex

The ctable package[I] for use with LaTeX2e

1 Purpose

The ctable package lets you easily typeset centered, captioned table and figure floats with optional footnotes. Both caption and footnotes will be forced within the width of the table.

If the width of the table is specified, then tabularx will be used to typeset it, and the X column specifier can be used. Otherwise tabular will be used.

This package defines the commands \ctable, \tnote and \tmark, as well as four \tabularnewline generating commands. The latter generate reasonable amounts of whitespace around horizontal rules and are also useful for tabulars outside this package.

Since the ctable package imports the array and booktabs packages, all commands from those packages are available as well.

Note that, in line with the comments that Simon Fear made describing his booktabs package, vertical rules for column separation can be produced with \ctable, but no provisions are made to have them make contact with horizontal rules.

2 Usage

\ctable

\ctable is called with 4 parameters, of which the first is optional:

```
\ctable[options]       % key=value,...
       {coldefs}       % for \begin{tabular}
       {foottable}     % zero or more \tnote commands (see below)
       {table lines}   % lines for the table
```

Options are given as key=value pairs, separated by comma's. Extra comma's, including one behind the last pair, don't hurt. Currently the following option keys have been defined:

[I] This file has version number v1.3, dated 2002/07/16.

caption	table caption
cap	for a short caption to go to the \tableofcontents
pos	float position, default: tbp
label	for \label
width	for tabularx; if absent, tabular will be used
figure	produce a figure float instead of a table float
botcap	put the caption at the bottom of the float instead of on top of it

The footnotes are placed under the table, without a rule. You therefore probably will want to use the \LL (last line) command if you use footnotes.

\tnote \tnote[label]{footnote text} places ^{label}footnote text under the table. Can only be used in the foottable parameter described above. The label is optional, the default label is a single *a*. For more detailed control, you can also replace this command with something like labeltext&footnotetext\NN.

\tmark \tmark[label] this command places the superscripted label in the table. It is equivalent with ^{label}. The label is optional, the default label is a single *a*.

The newline generating commands are a combination of \tabularnewline and zero or one of booktabs' \toprule, \midrule or \bottomrule. These combinations have been made, and short names have been defined, because source texts for complex tables often become very crowded:

\NN	Normal Newline, generates just a normal new line
\FL	First Line, generates a new line and a thick rule with some extra space under it
\ML	Middle Line: generates a new line and a thin rule with some extra space over and under it
\LL	Last Line: generates a new line and a thick rule with some extra space over it

These macros can be used outside \ctable constructs.

Finally, for completeness, here are some of booktabs' commands that may be useful:

\toprule	\toprule[<wd>] where <wd> is the optional thinkness of the rule
\midrule	\midrule[<wd>]
\bottomrule	\bottomrule[<wd>]
\cmidrule	\cmidrule[<wd>](<trim>){a-b} where <trim> can be r, l, or rl and the rule is drawn over columns a through b
\morecmidrules	\morecmidrules must be used to separate two successive cmidrules
\addlinespace	\addlinespace[<wd>] inserts extra space between rows
\specialrule	\specialrule{<wd>}{<abovespace>}{<belowspace>}

See the booktabs documentation for details.

3 Examples

3.1 Tables

Table 1 is an example taken from the related package threeparttable.sty by Donald Arseneau, with an extra footnote. It was typeset with:

```
\ctable[cap     = The Skewing Angles,
        caption = The Skewing Angles ($\beta$) for
                  $\fam0 Mu(H)+X_2$ and $\fam0 Mu(H)+HX$~\tmark,
        label   = tab:nowidth,
       ]
       {rlcc}
```

Table 1: The Skewing Angles (β) for $\mathrm{Mu(H)} + \mathrm{X_2}$ and $\mathrm{Mu(H)} + \mathrm{HX}$ [a]

	$\mathrm{H(Mu)} + \mathrm{F_2}$	$\mathrm{H(Mu)} + \mathrm{Cl_2}$
β(H)	80.9°[b]	83.2°
β(Mu)	86.7°	87.7°

[a] for the abstraction reaction, $\mathrm{Mu} + \mathrm{HX} \rightarrow \mathrm{MuH} + \mathrm{X}$.

[b] 1 degree $= \pi/180$ radians.

[c] this is a particularly long note, showing that footnotes are set in raggedright mode as we don't like hyphenation in table footnotes.

Table 2: Example with a specified width of 100mm

Example using tabularx			
	Multicolumn entry!	THREE	FOUR
one	The width of this column depends on the width of the table.[a]	three	Column four will act in the same way as column two, with the same width.

[a] footnotes are placed under the table

```
{\tnote{for the abstraction reaction,
        $\fam0 Mu+HX \rightarrow MuH+X$.}
 \tnote[b]{1 degree${} = \pi/180$ radians.}
 \tnote[c]{this is a particularly long note, showing that
           footnotes are set in raggedright mode as we don't like
           hyphenation in table footnotes.}
}{\FL
    &            & $\fam0 H(Mu)+F_2$     & $\fam0 H(Mu)+Cl_2$
  \ML
    &$\beta$(H)  & $80.9^\circ$\tmark[b] & $83.2^\circ$
  \NN
    &$\beta$(Mu) & $86.7^\circ$          & $87.7^\circ$
  \LL
}
```

Table 2 is an example with a width specification, taken from the `tabularx` documentation, with the vertical rules removed, and typeset with:

```
\ctable[caption = Example with a specified width of 100mm,
        width   = 100mm,
        pos     = t,
        label   = tab:width,
       ]
       {c>{\raggedright}Xc>{\raggedright}X}
```

```
{\tnote{footnotes are placed under the table}}
{\FL
   \multicolumn{4}{c}{Example using tabularx}
 \ML
   \multicolumn{2}{c}{Multicolumn entry!} & THREE & FOUR
 \ML
   one&
   The width of this column depends on the
   width of the table.\tmark &
   three&
   Column four will act in the same way as
   column two, with the same width.
 \LL
}
```

3.2 Figures

Figures, even single ones, are always put in tabular cells. This is not particularly handy for single pictures, but it eases the construction of arrays of pictures, including sub-captions, delineation, and spacing. Figure 1 shows a figure that has been produced with the `\ctable` command, in combination with `\usepackage{carom}`; it has been typeset with:

```
\ctable[caption=The di- and tri-bromobenzenes,
        botcap,
        figure,
       ]{ccc}{}{            \FL
 \bzdrv{1==Br;2==Br}&
 \bzdrv{1==Br;3==Br}&
 \bzdrv{1==Br;4==Br}        \NN
 1,2 & 1,3 & 1,4            \ML
 \bzdrv{1==Br;2==Br;3==Br}&
 \bzdrv{1==Br;2==Br;4==Br}&
 \bzdrv{1==Br;3==Br;5==Br}  \NN
 1,2,3 & 1,2,4 & 1,3,5      \LL
}
```

(The excessive whitespace at the left of the figure is caused by the bounding boxes generated by the *carom* package.)

4 Implementation

```
⟨*package⟩
\RequirePackage{keyval,array,tabularx,booktabs}

\def\NN{\tabularnewline}
\def\FL{\toprule}
\def\ML{\NN\midrule}
\def\LL{\NN\bottomrule}

\newcommand{\tnote}[2][a]{%
  \hbox{\@textsuperscript{\normalfont\textit{#1}}}&#2\NN}
```

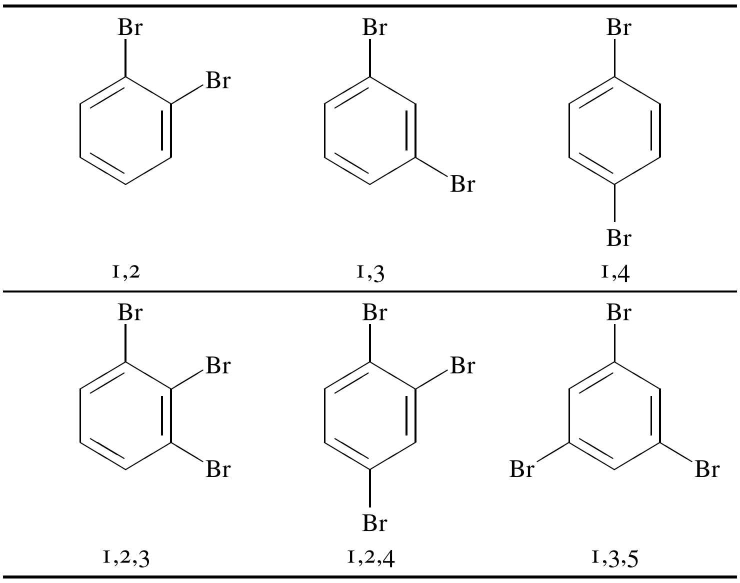


Figure 1: The di- and tri-bromobenzenes

```
\newcommand{\tmark}[1][a]{%
  \hbox{\@textsuperscript{\normalfont\textit{#1}}}}

```

Option setting commands from keyval. The table position (here, top, bottom, page) gets a special treatment, since LaTeX does not expand commands there. So instead of putting things like `tbp` in a command like `\ctblbegin` we put `\begin{table}[tbp]` in it.

```
\define@key{ctbl}{caption} {\def\ctblcaption{#1}}
\define@key{ctbl}{cap}     {\def\ctblcap    {#1}}
\define@key{ctbl}{label}   {\def\ctbllabel  {#1}}
\define@key{ctbl}{pos}     {\def\ctblbegin  {\ctblbeg[#1]}}
\define@key{ctbl}{width}   {\def\ctblwidth  {#1}}
\define@key{ctbl}{botcap}[]{\def\ctblbotcap {1}}
\define@key{ctbl}{figure}[]{\def\ctblbeg{\begin{figure}}%
                            \def\ctblend{\end{figure}}}

\def\ctblCaption{
  \ifx\ctblcap\empty%
    \caption{\label{\ctbllabel}\ctblcaption}%
  \else
    \caption[\ctblcap]{\label{\ctbllabel}\ctblcaption}
  \fi
}
```

Allocate a box register for use by the `\ctable` command.

```
\newbox\ctbl@tabel

\newcommand{\ctable}[4][]{%
  \def\ctblcaption {}%
  \def\ctblcap     {}%
  \def\ctbllabel   {}%
  \def\ctblbeg     {\begin{table}}%
  \def\ctblbegin   {\ctblbeg}%
  \def\ctblend     {\end{table}}%
  \def\ctblwidth   {}%
  \def\ctblbotcap  {}%
  \setkeys{ctbl}{#1}%
  % save the table contents in a box, so we can determine its width:
  \sbox\ctbl@tabel{%
    \ifx\ctblwidth\empty
      \begin{tabular}{#2}%
        #4%
      \end{tabular}%
    \else
      \begin{tabularx}{\ctblwidth}{#2}%
        #4%
      \end{tabularx}%
    \fi
  }%
```

\ctblbegin is now defined as something like \begin{table}[tbp].

```
  \ctblbegin
    \begin{center}%
      \begin{minipage}{\wd\ctbl@tabel}%
        \ifx\ctblbotcap\empty\ctblCaption\vspace{2ex}\fi
        \usebox\ctbl@tabel % insert the tabular
        \def\ctblfootnotes{#3}%
        \ifx#3\empty\else % append footnotes, if any
           \\%
           \begin{tabularx}{\hsize}{r@{\,}>{\footnotesize\raggedright}X}%
           #3%
           \end{tabularx}%
        \fi
        \ifx\ctblbotcap\empty\else\ctblCaption\fi
      \end{minipage}%
    \end{center}%
  \ctblend%
}
⟨/package⟩
```

mk – a LATEX maker

Wybo Dekker
wybo@servalys.nl

1 Introduction

MK is a Perl script that, in close collaboration with VPP [1] is helpful in the cyclic process of editing, viewing, and printing a LATEX document. Having an existing[2] LATEX document, say *main.tex*, you run MK on it by typing:

```
mk main
```

or, since *main* happens to be MK's default filename:

```
mk
```

Now, if *main.tex* is a valid LATEX source, MK compiles it, including any table of contents, indices, bibliography references, included files, and so on and VPP takes over and displays the resulting POSTSCRIPT or PDF output. When you leave the viewer you will see a prompt:

```
vpp command (h for help):
```

If you were satisfied with the displayed POSTSCRIPT or PDF output, you can now decide to print all or part of your document (see the section 'Page selection'), or you can simply quit by typing 'q'. On the other hand, if you decide that you want to change the source and have another try, you can edit the source by typing 'e' to get back to MK and (re)edit your source. After saving your work and leaving your editor, another compilation and display will be performed, based on the new source.

Essentially, MK uses the MAKE utility for the management of file dependencies, TEXI2PDF for compilation, GV (or any other viewer you like) for viewing, and CONTEXT's TEXEXEC for printing in various formats.

Currently, MK is only available for UNIX, but adaptation for MSDOS/WINDOWS should not be too complicated, as all software needed, including MAKE, is available on those operating systems.

2 How it works

When working on a LATEX document, the main activities — apart from thinking — consist of editing the sources, compiling with (PDF)LATEX, viewing the result (either a POSTSCRIPT or PDF file or LATEX errormessages) and, perhaps, printing the document or parts of it.

A LATEX document, as well as any other TEX document, may need several passes of compilation in order to fulfill all cross references and bibliography references, fix longtable calculations, and build the indices. When you compile manually, you'll have to keep track of the often abundant messages of LATEX to see if another compilation is needed. Most of that work can be taken out of your hands by using TEXI2PDF (or TEXI2DVI.) That is exactly what MK does.

[1] VPP (short for View and Print PostScript/PDF) is the subject of another article in this issue.

[2] We shall later explain what happens if you try to edit a non-existing file (see the section 'Locating the source').

However, although TEXI2PDF does look in the current directory for `\include`d and `\input`ed files, it does not keep track of other files that you may have edited, such as style files, bibliography files, fontfiles, and so on, either in the current or in other directories. For that purpose, a recent contribution to CTAN by Tong Sun and Chris Beggy comes in handy: a makefile for MAKE that takes care of all of this, in cooperation with TEXI2PDF. However, this makefile needs to be told on what files your sources depend (apart from included files in the current directory.)

This problem is solved by the recent appearance of a new option for TeX: the **–recorder** option. This option tells TeX to maintain a *.fls* file that logs all file dependencies that TeX finds in the sources. This is how the start of the current document's *.fls* file, *main.fls*, looks like:

```
PWD /home/wybo/CVSWORK/mk
INPUT /tex/texmf/web2c/pdflatex.fmt
INPUT /tex/texmf/pdftex/config/pdftex.cfg
INPUT /home/wybo/CVSWORK/mk/main.tex
OUTPUT main.log
INPUT /texlive/texmf/tex/latex/base/article.cls
[ 110 similar lines follow... ]
```

Now here is how MK works (supposing *main.tex* to be the source):

1. if there is no file *main.fls* or if it is older than the source, LATEX is run to generate it.
2. *main.fls* is scanned for lines starting with INPUT and the files on those lines are saved for MAKE.
3. MAKE is executed.
4. if an error occurred, the log file is displayed, starting at the error location, skipping irrelevant lines, and stopping at most 20 lines later. The error message and its line number in the source are highlighted in color. The line number is remembered for the editor to start at. The user is finally asked whether he wants to quit, or to edit the source and recycle from 1. on.
5. if no error occurred, the PDF or POSTSCRIPT output is displayed, using VPP.
6. after the user has left the viewer (normally with 'q' or 'control-q') the user is asked (still running VPP) whether he wants to quit, or (re-)edit the source, or to print (parts of) the document.

3 Page selection

As said in the introduction, after a successful compilation and display of the resulting PDF or POSTSCRIPT output, the user is prompted with:

```
vpp command (h for help):
```

upon typing 'h' VPP displays examples of possible commands:

```
Examples of print commands:
  5       to print page 5
  5-      to print pages 5 through the end
  5-7     to print pages 5, 6 and 7
  -7      to print the first 7 pages
  5-7,19- to print pages 5, 6, 7 and 19 through the end
  a       to print the whole document
  a x3    to print 3 copies of the document
  x3      the same
  5 x3    to print 3 copies of page 5
  t       print the whole document twosided
  t 2-    print twosided starting at page 2
  b       to print the whole document as an a5 size booklet
  b -12   to print the first 12 pages as an a5 size booklet
Other commands:
  e       edit the tex source and rerun mk
  h       display this help
  ?       display this help
  q       quit
vpp command (h for help):
```

With these examples, no further explanation should be necessary, except that, when twosided ('t') or booklet ('b') printing is selected, printing will be performed in two shifts, one for the front side and one for the backside. Between the shifts, another prompt appears:

```
printer ready?  then turn stack and type return
```

You will have to arrange your printer such that, with the printed sides up, the first page printed will be at the bottom of the stack, and the last page printed will be on top. Normally you will then have your output come out the back of your printer. 'Turn the stack' then means: rotate it over the long side of the paper and feed it back into the printer for the other side to be printed.

For further information on VPP, look in its manpage by typing

```
vpp -h
```

or read the article on VPP elsewhere in this issue.

4 Locating the source

MK locates the LATEX source in several steps:

1. If you supply no arguments, the file *main.tex* in the current directory is assumed.
2. If you supply an argument (say *myfile,*) MK adds a *.tex* extension if it isn't there and looks for *myfile.tex* in the current directory.
3. if *myfile.tex* is not found in the current directory, MK looks in the 'alternate directory' (say */Documents*) if you have defined one (see the section 'RC-files').
4. if the source was not found in /Documents, MK thinks that you may have a subdirectory *myfile* in /Documents where the source may live under the name *main.tex*
5. if that file is not there, MK now concludes that the source does not yet exist and reports this, telling at the saem time which files have been tried.
6. if you have defined a template file (see the section 'RC-files'), MK now gives you the opportunity to create a new LATEX source from that template. If you confirm MK's question, MK copies the template to the filename you supplied (or *main.tex* if you did not) and starts your editor with the newly created file.
7. finally, if all the above did not lead to a source file, MK dies.

Table 1 MK options

	Options	Short	Defaults:
	--batch=*selection*	**-b**	
	--clean	**-c**	
	--Clean	**-C**	
⋆	**--print**	**-pr**	true
⋆	**--view**	**-vi**	true
⋆	**--ps**	**-ps**	false
⋆	**--quiet**	**-q**	true
	--rc=*rcfile*	**-r**	
	--edit=*file*	**-e**	
	--help	**-h**	
	--version	**-ve**	

5 Options

MK comes with several options. Table 1 shows an overview. Options are shown in logically identical pairs, with the full version in the first column and the minimum shorthand (without the parameters) in the second. Options marked with a star are boolean options. Default values are shown in the last column. You can set boolean options to false by prefixing the option with 'no', for example: **--noquiet** or **-noq**.

Before evaluating any options, MK will try to read a system rc-file, a user rc-file, and, finally an rc-file in the current directory. The default values for ⋆-marked options and for string options can be set in these files. See the section 'RC-files' for more information.

You can also set option defaults in an alias. For example:

```
alias mk='mk -noquiet'
```

--help Prints help information and lets you type 'm' to display the complete man page or anything else to quit.

--version Prints name and (CVS-)version and then quits.

--quiet Suppresses messages about the progress MK is making. This is the default.

--rc *rc-file* Read specified rc-file before processing. The contents of the rc-file may override options specified before the **--rc** option, therefore it is a good idea to have the habit of specifying the **--rc** option first.

--batch *printing command string* Prevents the **--print** option to interrogate the user about pages to be printed. Instead the document is printed according to the mandatory *printing command string*. Also sets viewing off. Thus the command

```
mk -batch '2-3 x3' test
```

prints 3 copies of pages 2 and 3 of *test.tex*, without viewing.

--clean Clean up (remove) all unnecessary files generated by LATEX and BIBTEX except for the PDF or POSTSCRIPT files.

––Clean Clean up (remove) all unnecessary files generated by LATEX and BIBTEX including the PDF or POSTSCRIPT files.

––print Present the print prompt. This is the default. This option is normally used to suppress the print prompt, for example when using MK from other scripts that generate LATEX documents that have only to be displayed or stored without even being displayed.

––ps Generate POSTSCRIPT version of document. The default is to generate a PDF document.

––view Run the file viewer. This is the default. This option is normally used to suppress starting the viewer, for example when using MK from other scripts that generate LATEX documents that have only to be printed.

––edit *file* Normally, MK lets you edit the main source file, but here you can specify another file to be edited instead. This is useful, for example, if you are are fixing a style file or another input file.

6 RC-files and customization

Unless the environment variable NORC has been set, three rc-files are executed, if they exist, before reading the command line options, in the following order:

1. `/etc/mkrc`: the system rc-file
2. `$HOME/.mkrc`: the user rc-file
3. `./.mkrc`: the local rc-file

You can use these rc-files to set the default values for the options, by setting the Perl variable named after the long version of the options. For example:

```
$quiet=1; # run in quiet mode
```

So if you usually like MK to work quietly, you can indicate so in your rc-file and change your mind in some cases by using the **––noquiet** (or perhaps **–noq**) option.

Other variables that can be set in the rc-files, and their default values, are:

`$skip_pattern = '';` can be set to a file wildcard pattern. Files matching this pattern on which the LATEX source file may depend will not be checked for changes. For example, if you use a write-protected TEX-tree in the directory *texlive* it makes sense to set `$skip_pattern='/^\/texlive/';`

`$altdir = '';` If `$altdir` is non-empty and a file to be compiled does not exist in the current directory, it will be given another try after prefixing it with the contents of `$altdir`. So if you like to have your LATEX file in */Documents/myfile.tex* you can set `$altdir` to `/Documents` and run MK from any directory with:

```
mk myfile
```

However, a directory like */Documents* does not make much sense if many of your LATEX documents do not consist of a single file, but are constituted of an ensemble of a main LATEX source and one or more `\include`d and `\input`ed files such as graphics. You will then probably prefer to have s subdirectory in */Documents* for every LATEX document.

Therefore, if MK does not find *myfile.tex* in the alternate directory, it will assume that *myfile* is a subdirectory with a main LATEX source in it, called *main.tex*.

`$default = 'main';` This is the default for the basename of your LATEX document.

`$template = '';` Tells MK to give the opportunity to create a copy of this file when a non-existent source is requested.

scripting

Wybo Dekker
wybo@servalys.nl

vpp – View and (selectively) Print PDF and POSTSCRIPT

1 Introduction

VPP displays a PDF document or a POSTSCRIPT document (after conversion to PDF) using GV, XPDF, or ACROBAT READER (in that order of choice), or perhaps another PDF viewer. One can use the viewer to print the document but, alternatively, leave the viewer and use VPP's facilities to print selected pages to a one- or two-sided hardcopy or an A5-booklet: see the section 'Page selection' for the details.

If *file* if not found, VPP looks for *file.pdf* or *file.ps*. If both are found, VPP dies, saying that it can't make a choice, if one of the two is found it is used instead of *file*.

If *file* lacks, standard input is used.

2 How it works

For the printing of PDF files, VPP makes use of CONTEXT's TEXEXEC utility, except when you simply print one-sided and without scaling: in that case VPP just offers your PDF file to the print spooler.

For twosided printing, VPP first selects the odd pages into a temporary PDF file and sends it to the print spooler, and then prompts the user for an <Enter> key-press, selects the even pages and prints them:

```
# odd pages:                    # even pages
texexec --result=tmp.pdf \      texexec --result=tmp.pdf \
        --pdfselect \                   --pdfselect \
        --selection=odd \               --selection=even \
        input.pdf \                     input.pdf
lpr tmp.pdf                     lpr tmp.pdf
```

For a booklet, VPP first lets TEXEXEC rearrange and scale the pages, and then prints both sides as before:

```
texexec --result=tmp.pdf \      texexec --result=tmp.pdf \
        --pdfarrange \                  --pdfarrange \
        --paper=A5A4 \                  --paper=A5A4 \
        --print=2UP \                   --print=2UP \
        --pages=odd \                   --pages=even \
        input.pdf                       input.pdf
lpr tmp.pdf                     lpr tmp.pdf
```

3 Options

VPP comes with several options. Table 1 shows an overview.

Options are shown in logically identical pairs, with the full version in the first column and the minimum shorthand (without the parameters) in the second. Options marked with

Table 1 VPP options

	Options	Short	Defaults:
	--batch=*selection*	**-b**	
⋆	**--print**	**-pr**	true
⋆	**--view**	**-vi**	true
⋆	**--quiet**	**-q**	true
	--rc=*rcfile*	**-r**	
	--help	**-h**	
	--version	**-ve**	

a star are boolean options. Default values are shown in the last column. You can set boolean options to false by prefixing the option with 'no', for example: **--noquiet** or **-noq**.

Before evaluating any options, VPP will try to read a system rc-file, a user rc-file, and, finally an rc-file in the current directory. The default values for ⋆-marked options and for string options can be set in these files. See the section 'RC-files' for more information.

You can also set option defaults in an alias. For example:

```
alias vpp='vpp -noquiet'
```

--help Prints help information and lets you type 'm' to display the complete man page or anything else to quit.

--version Prints name and (CVS-)version and then quits.

--quiet Suppresses messages about the progress VPP is making. This is the default.

--rc *rc-file* Read specified rc-file before processing. The contents of the rc-file may override options specified before the **--rc** option, therefore it is a good idea to have the habit of specifying the **--rc** option first.

--batch *printing command string* Prevents the **--print** option to interrogate the user about pages to be printed. Instead the document is printed according to the mandatory *printing command string*. Also sets viewing off. Thus the command

```
vpp -batch '2-3 x3' test.pdf
```

prints 3 copies of pages 2 and 3 of *test.pdf*, without viewing.

--print Present the print prompt. This is the default. This option is normally used to suppress the print prompt, for example when using VPP from other scripts that generate PDF or POSTSCRIPT documents that have only to be displayed or printed without even being displayed.

--view Run the file viewer. This is the default. This option is normally used to suppress starting the viewer, for example when using VPP from other scripts that generate PDF or POSTSCRIPT documents that have only to be printed.

4 Page selection

When you select the **--print** option, and you did not also use the **--batch** option, VPP interrogates you about the pages you want to print. To that end the following prompt appears:

```
vpp command (h for help):
```

upon typing ‘h’ VPP displays examples of possible commands:

```
Examples of print commands:
  5       to print page 5
  5-      to print pages 5 through the end
  5-7     to print pages 5, 6 and 7
  -7      to print the first 7 pages
  5-7,19- to print pages 5, 6, 7 and 19 through the end
  a       to print the whole document
  a x3    to print 3 copies of the document
  x3      the same
  5 x3    to print 3 copies of page 5
  t       print the whole document twosided
  t 2-    print twosided starting at page 2
  b       to print the whole document as an a5 size booklet
  b -12   to print the first 12 pages as an a5 size booklet
Other commands:
  h       display this help
  ?       display this help
  q       quit
vpp command (h for help):
```

With these examples, no further explanation should be necessary, except that, when twosided (‘t’) or booklet (‘b’) printing is selected, printing will be performed in two shifts, one for the front side and one for the backside. Between the shifts, another prompt appears:

```
printer ready?  then turn stack and type return
```

You will have to arrange your printer such that, with the printed sides up, the first page printed will be at the bottom of the stack, and the last page printed will be on top. Normally you will then have your output come out the back of your printer. ‘Turn the stack’ then means: rotate it over the long side of the paper and feed it back into the printer for the other side to be printed.

5 RC-files and customization

Unless the environment variable NORC has been set, three rc-files are executed, if they exist, before reading the command line options, in the following order:

1. `/etc/vpprc`: the system rc-file
2. `$HOME/.vpprc`: the user rc-file
3. `./.vpprc`: the local rc-file

You can use these rc-files to set the default values for the options, by setting the Perl variable named after the long version of the options. For example:

```
$quiet=1; # run in quiet mode
```

So if you usually like VPP to work quietly, you can indicate so in your rc-file and change your mind in some cases by using the **--noquiet** (or perhaps **-noq**) option.

One other variable that can be set in the rc-files is:

```
$viewer = 'gv'; # set the viewer
```

GV is the default viewer for both POSTSCRIPT and PDF. If you normally use PDF and have set, say, xpdf as your viewer, you would get in trouble when handling a POSTSCRIPT file. In such cases, therefore, VPP tries to switch to GV.

6 Examples

Since VPP can read from standard input, it can be used to print (parts of) manpages:

```
man -t ls | vpp
```

If you don't need a preview, because you have seen the man page already, you can print it immediately as an A5 booklet with:

```
man -t ls | vpp -batch=b
```

metapost

Shifted bullets in graphs with METAPOST

Frans Goddijn
frans@goddijn.com
Karel H Wesseling
k.h.wesseling@planet.nl

abstract
With METAPOST fully integrated in CONTEXT using this graphic language has become convenient. When we tried to use John Hobby's "Graph.mp" package, however, it turned out to require extra initializations and to produce unacceptable, shifted data graphs. Solutions to both problems are given.

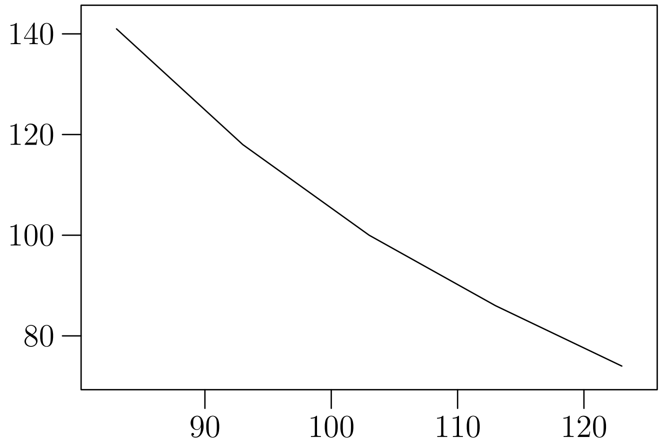
Introduction

METAPOST has become fun for these users when it became integrated in CONTEXT, which was the trigger for this exercise, but the same can be done in LATEX [1]. METAPOST is known for the high precision of its graphics output and for its perfect scalability. Since this sounds promising also for the presentation of scientific data in graphical form we looked for a package that would ease the plotting of data files. This was found in the "graph" package written by John Hobby [2], the author of METAPOST. In using this package we met with two problems: one, the lacking of proper initializations and two, the inaccurate placement of bullets at given coordinates.

Initialization

The inclusion of a METAPOST graphics specification in a CONTEXT document is simple and below is a code fragment that produces an XY–plot and places it as a graphic in this document.

```
\startuseMPgraphic{plot}
  draw begingraph(3in,2in);
    gdraw "bdoffset.d";
    endgraph;
\stopuseMPgraphic
\midaligned{\useMPgraphic{plot}}
```



With `\startuseMPgraphic{plot}` ConTeXt is told that all instructions until the `\stopuseMPgraphic` are to be understood as MetaPost–speak. `draw` is a MetaPost instruction and `gdraw` is one included in Hobby's "graph" package which lives on top of MetaPost. The file `bdoffset.d` is a small data file arranged as:

```
83 141
93 118
103 100
113 86
123 74 <EOF>
```

This file must not have any extra empty lines (or all hell breaks loose). As a reminder, therefore, we placed an unnecessary `<EOF>` right after the final datum. The "graph" package accepts extra characters on a line, but no extra lines.

If you would try this program, the reason that it doesn't work right away is that the "graph" package has not been input. This would usually be accomplished by an instruction

```
\startMPinclusions input graph; \stopMPinclusions
```

placed near the top between the start/stop of `useMPgraphic`. Since extra initializations appeared also needed, however, Hans Hagen[1] wrote the following little module that must be input for use (`\usemodule[graph]`) in the preamble area of the source file, i.e. before `\starttext`:

```
\forceMPTEXgraphictrue
\appendtoks
  initialize_numbers; % assure that pseudo typesetting is set up
  input graph;        % load the graph package
  Autoform:="@g";     % change the % template char into @
\to \MPinitializations
```

Place these instructions in a file named `m-graph.tex` and the graph package is input, properly initialized, and ready for work. Note that the file `m-graph.tex` is input with `\usemodule[graph]`. When ConTeXt reads `\usemodule` it inserts `m-` and appends `.tex` to the name automatically to find the correct file. Be sure to also place Hobby's `graph.mp` in the same directory or at least where it can be found by ConTeXt. Using a rather recent version of ConTeXt (see "References") and this module, compile again, and the data plot shown above should result.

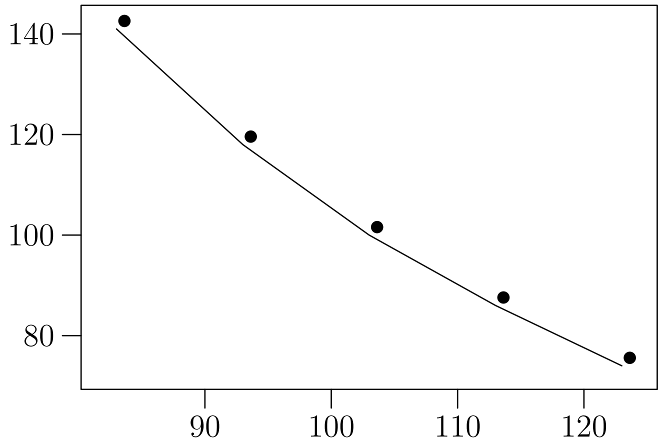
Shifted bullets

In the data plot the data pairs are plotted interconnected with straight lines. Since the curve is featureless the positions of the individual data points are barely discernable. Let us try to mark them more clearly. The manual for the "graph" package offers a solution. Simply replot the data but this time as bullets with the instruction

```
    gdraw "bdoffset.d" plot btex$\bullet$etex
```

The `btex...etex` pair signifies to MetaPost that the instructions in between should be interpreted as TeX–typeset instructions. Thus, within the ConTeXt document we first instruct ConTeXt to look at a set of text lines as MetaPost and within these lines MetaPost is asked to consider some words as instructions to TeX, which is why you need a recent version ConTeXt. Still, the system works and the result is the following graphic:

[1] of Pragma–ADE, the author of ConTeXt.



The dots don't fall on the line. Our first thought was that we did something wrong and our first action was to ask for help. Our second action was to check the graphic made with similar instructions in the "graph" manual. By tearing out the manual's page on which Fig. 2 is plotted and placing it over Fig. 1 such that the tick marks match we observed that equally in the manual the dots appear shifted towards the upper right. It turns out that TeX positions each glyph with its reference point, not its midpoint, at the specified coordinates [3]. The reference point of the bullet is at the lower left of its box:[2] ▣. Thus, neither TeX nor we did wrong but the instruction is wrong.

The first solution suggested is to shift the bullet in opposite direction by replacing the TeX instruction `$\bullet$` with:

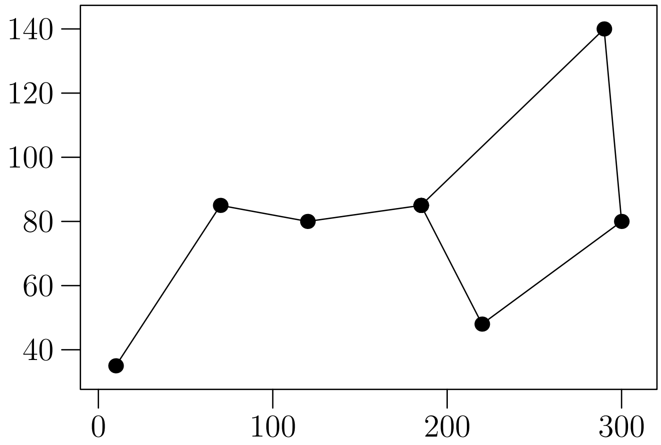
```
\kern-1mm\raisebox{-1mm}[0mm][0mm]{$\bullet$}
```

which should shift the dot 1 mm to the left and 1 mm down, but doesn't. TeX within MetaPost within Context forgets the original coordinates somewhere on the way and nothing is output. Since we hard–program shifts in mm for glyphs that could vary in dimension with the symbol specified it is not a general purpose solution either. Yet, it might have worked.

A solution which stays within MetaPost came from the author of MetaFun [4]. It declares and defines the bullet as a MetaPost picture and plots the picture repeatedly at each coordinate specified. The full text of the example becomes:

```
\startuseMPgraphic{buldot}
  picture Buldot;
    Buldot:=image(draw origin withpen pencircle scaled 2mm;);
  draw begingraph(3in,2in);
    gdraw "other.d";
    gdraw "other.d" plot Buldot;
    endgraph;
\stopuseMPgraphic
\midaligned{\useMPgraphic{buldot}}
```

[2] This can be observed in Context by typing `\ruledhbox{$\bullet$}`



The problem is solved. The ‘picture’, of course, could be anything, like stars or squares or triangles or even doughnuts, as long as its midpoint is defined at the origin. And yes, we did change the data points to something more recognizable (a big dipper).

Acknowledgement

We would like to express our appreciation to the respondents to our questions, they know who they are, that offered help and helped to a nice solution, all within the timespan of one workday.

References

1. Popineau F: MetaPost pratique. Cahiers GUTenberg n^{o} 41 — Novembre 2001.
2. Hobby J D: Drawings graphs with MetaPost.
   `\texmf\doc\metapost\base\mpgraph.pdf`
3. Knuth D E: The TEXbook (Chapter 11). Addison Wesley 1970.
4. Hagen H: Metafun.
   `\texmf\doc\context\manuals\metafun-p.pdf`

The directory paths above are for the TEXLive CDROM set, issue nr 7, June 2002, Volume 1. The MetaFun manual and the ConTeXt software are also available from `http://www.pragma-ade.com/`

metapost

A letterhead in CONTEXT

Karel H Wesseling
k.h.wesseling@planet.nl

abstract
For years I have used a home–made logo in PICTEX within LATEX, together with name and address as letterhead. Separate versions for myself and my wife were pre–printed on an HP 300 DPI Laserjet. With METAPOST fully integrated in CONTEXT, we decided to convert to METAPOST and print the letterhead with each letter automatically. I used the versatile CONTEXT layer mechanism and the mode option.

keywords
non–guru, CONTEXT, METAPOST, layers, mode command

Introduction

Years ago, I used PICTEX within LATEX to produce a personalized letterhead for the private correspondence of my wife and myself. With new versions of TEX and new printers coming our way, however, more maintenance appeared needed than I liked. We started with PCTEX but moved to 4TEX–the–magnificent, no longer available, as soon as we found out about it. We started with a 300 DPI HP Laserjet but upgraded to 600 DPI and obtained DeskJet's with 1200 by 600 DPI. With each change, almost, my letterhead moved an inch or less up or down, left or right, on the page. With a non–guru approach to LATEX, meaning feeling unable to write a style—now called class— I had a hard time to reposition. I often achieved what was wanted but never quite understood how exactly. Conversion to the DeskJet was given up as too much work all over again. Even more difficult appeared the typesetting of the letterhead and the letter text in one go. So we had a two step process, one: to print the letterhead and two: to print the letter itself on the preprinted paper.

Then, a few years ago I converted to CONTEXT which has both PICTEX [2] and METAPOST [3] integrated in the language and in addition offers the layer mechanism for easy positioning and scaling of layers of printed material on the paper sheet, and modes to compile different printed outputs from the same source text using a switch on the command line. Since everything also seemed somewhat easier to accomplish in the CONTEXT language we now wanted everything integrated and automated. What follows is a narrative on how this was accomplished.

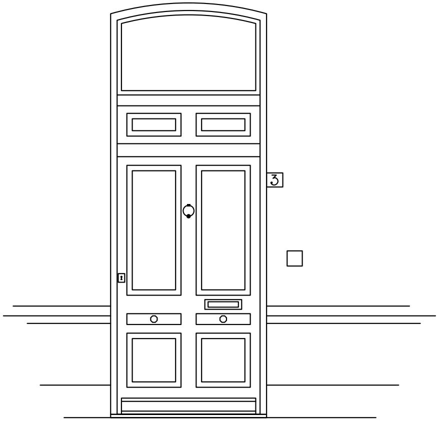
The logo in METAPOST

We had a logo as a PICTEX graph representing our front door as a design sketch, as we eventually would like to have it made. Done in PICTEX it turned out to be difficult to join an arc to a straight line with precision and the relative positioning of separate lines with respect to rectangles, for example, seemed slightly inexact. Whereas the arc was built of cmr5 sized dots, the straight lines seemed drawn more like a line and their thickness was hard to match to the dots in the arc. Arrows at the end of a line never seemed to be part of it but always appeared stuck on. This was not noticeable in a small size graph at 300 DPI but enlarging the graphic on screen only a few times revealed these inaccuracies. Thus, although PICTEX facilitated the drawing of very nice, professional looking, scientific diagrams and had an exceptionally clear and useful manual, when we were assured that METAPOST [1] offered greater precision, we decided to give it a try.

In the mean time the door was built and deviated somewhat from the original logo drawing, urging the need for a new drawing. The major features of the door were measured and rounded to cm, and put into MetaPost code. MetaPost has bp (big point) as its default internal unit and the door dimensions in cm can be entered without any dimensional conversion to give a drawing that fits an A4 sheet. What follows are a few example lines of code from a file `mpdeur.tex` that produce the outline, the letter box and a cassette.

```
% output=pdftex
\noheaderandfooterlines
\starttext
\ \  % print something (a space) or else no output
\definelayer[mpdeur][position=no] % door in metapost
\setupbackgrounds[page][background=mpdeur]
\setlayer[mpdeur][x=1mm,y=1mm]{% position 1 mm down and left
\startMPcode
   pickup pencircle scaled 1;
% kozijn buiten rand
   draw (  0,  0) -- (144,  0) -- (144,370) &
        (144,370) .. ( 72,380) .. (  0,370) &
        (  0,370) -- cycle;
% brievenbus rechts buiten en binnenrand
   draw ( 87, 97) -- (121, 97) -- (121,106) -- ( 87,106) -- cycle;
   draw ( 90, 99) -- (118, 99) -- (118,104) -- ( 90,104) -- cycle;
% bovenste cassette links  buiten en binnenrand
   draw ( 15,110) -- ( 65,110) -- ( 65,230) -- ( 15,230) -- cycle;
   draw ( 20,115) -- ( 60,115) -- ( 60,225) -- ( 20,225) -- cycle;
% etc, etc.
% downsizing for this manuscript, different from letter logo
   currentpicture:=currentpicture scaled .5;
\stopMPcode
} % end of \setlayer
\stoptext
```

Compile this with ConTeXt[1] and everything is taken care of. The door logo is output as a `mpdeur.pdf` one page graphic ready for inclusion in a letter. With the full code the door graphic ('deur' is Dutch for door) is shown immediately below at half the size.



[1] Type `texexec mpdeur`

The picture above follows the text without empty space, which is not pleasing to see, but shows the positioning. Normally in CONTEXT one would correct this joined placement with `\startlinecorrection`. The style of the door is in tune with what was common in The Hague between 1850 and 1870, although the letter box has todays mandatory dimensions.

A few words on the CONTEXT layer instructions surrounding the METAPOST graphic. Ordinary text, when typeset—like the door graphic above— is placed by TEX beginning top left in the *text* area. The text area begins about an inch from the top and an inch from the left but can be setup (with `\setuplayout[options]`). I prefer not to touch the settings of the original layout. Positioning of the door–logo should still be independent of the text area on the page. This calls for a CONTEXT layered layout:

```
\definelayer[mpdeur][position=no]
\setupbackgrounds[page][background=mpdeur]
\setlayer[mpdeur][x=1mm,y=1mm]{%
  \startMPcode
    ...
  \stopMPcode
} % end of \setlayer
```

First we define a layer with the logical name (`mpdeur`) having absolute positioning (`\position=no`) with respect to the upper left of the page, then we setup the background for a page printed with this layer, then we set the layer to a “value” which is here the METAPOST graphic embraced by `\startMPcode ... \stopMPcode`. The picture is located 1 mm to the lower right from the upper left corner of the page. This distance could have been made 0 mm but then the feeling that parts might have been printed off paper is hard to suppress. After compilation with CONTEXT we obtain a page `mpdeur.pdf` showing the background and viewable with Acrobat. But the picture, although useable as a full page background, is much too big for a logo, and must be downsized.

Placement of the logo in the letterhead

With the door drawn to the far upper left of the paper it can now be scaled and placed anywhere on a sheet. We decide to have the logo towards the upper left, followed by name and address. The logo placement is achieved as follows:

```
% output=pdftex
\noheaderandfooterlines
\starttext
\definelayer[logohead][position=no]
\setupbackgrounds[page][background=logohead]
\setlayer[logohead][x=1.8cm,y=2cm]{%
  \externalfigure[mpdeur.pdf][scale=220]
} % end of \setlayer
\stoptext
```

The layer calls are the same as before but the name of the layer is now `logohead`. To achieve a pleasing layout we had to play a little with the scale and the position of the door graphic. This and reuse for other purposes are also the reason for including the graphic indirectly via a PDF file. The page number is suppressed with `\noheaderandfooterlines`.

Adding name and address

Next is to add the name and address of the sender. Both my wife and I want a separate letterhead. It seemed easiest if this could be specified on a command line perhaps as follows:

```
makeletter nameofletter withhead
```

A true windows panel with a name entering box and a series of buttons to choose the head type would perhaps have been nicer because then the exact name options of the `withhead` need not be remembered. Using Windows Commander, however, a small explanatory file could be placed in the same directory for counseling and be visibly present. Two batch files were made, one to start a new letter (`ML.BAT`) and one to edit an existing letter (`EL.BAT`). The first is:

```
: enter on a command line "ml nameofletter [karel|hanny]"
copy vb.tex %1.tex
ed.exe d:\mklt\%1.tex
texexec --mode=%2 %1
```

File `vb.tex` is a letter template that is copied to another name. We discuss `vb.tex` later. The file `ed.exe` is the PC–Write editor that I still prefer. When the letter has been prepared it must be compiled and this is done with a very versatile program `texexec` within the ConTeXt system that understands and is able to pass a mode to the ConTeXt compiler. The command line specification `--mode=karel` makes sure that my letterhead is typeset, `--mode=hanny` specifies hers. When I am ready to write my ninth letter to the city counsel all I do is type on the command line:

```
ml benw09 karel
```

Here is how the specified mode is used in the `vb.tex` letter template:

```
% output=pdftex
\noheaderandfooterlines
\setupbodyfont[12pt]
\starttext
\definelayer[logohead][position=no]
\setupbackgrounds[page][background=logohead]
\setlayer[logohead][x=1.8cm,y=2cm]{%
  \externalfigure[mpdeur.pdf][scale=220]
} % end of first \setlayer
\setlayer[logohead][x=4.4cm,y=2cm]{%
  \doifmode{karel}
    {Karel~H~Wesseling,~~Nieuwe
      Schoolstraat~3,~~2514~HT~Den~Haag}
  \doifmode{hanny}
    {J~W~M~Wesseling||Gommers,~~Nieuwe
      Schoolstraat~3,~~2514~HT~Den Haag}
} % end of second \setlayer
\stoptext
```

When ConTeXt scans the text it compares the command–line specified mode with the name within the first braces pair of `\doifmode` and if equal carries out the instructions between the next braces. This could have been accomplished instead with `\startmode[karel] Karel~H~Wes... \stopmode` which construct I prefer when more than one or two actions are to be taken. In the same `logohead` layer at the specified position the name and address is typeset in 12pt size CMR. It is not necessary to specify a new layer, just add the text to the layer already specified. Positioning of text or figures in a ConTeXt layer, thus, is not left to TeX but done by the designer.

Adding the signature

Although it is good practice to sign the letter manually, if only because it makes one scan the text briefly, with letters sent over the internet this is only possible when the letter is first printed, then manually signed, then scanned, then transmitted. This seemed a few steps too many. The thought then occurred that a signature could be written on a blank sheet, scanned, turned into a picture and added preferably automatically at the end.

The signature was made and filed as `kh.png`. Portable Network Graphics, PNG, is one of the graphic formats preferred by CONTEXT. The signature is added to the letter only when the mode is `karel`:

```
\useexternalfigure[karelsign][kh][width=.2\textwidth]
\starttext
....
\blank[2*big]
% Here the signature
% ------------------
\doifmode{karel}{\externalfigure[karelsign]}
\stoptext
```

With `\useexternalfigure` the logical name `karelsign` is created and linked with a file name `kh.png` of which the extension need not be given, but not all formats are acceptable to CONTEXT. The width is also set and CONTEXT scales both figure dimensions proportionally for the width to fit a small fraction of the text width. The `\blank` instruction creates some room between the "greetings" and the signature. This signature inclusion has made me realize how easy it has become to email someone a letter under a forged signature.

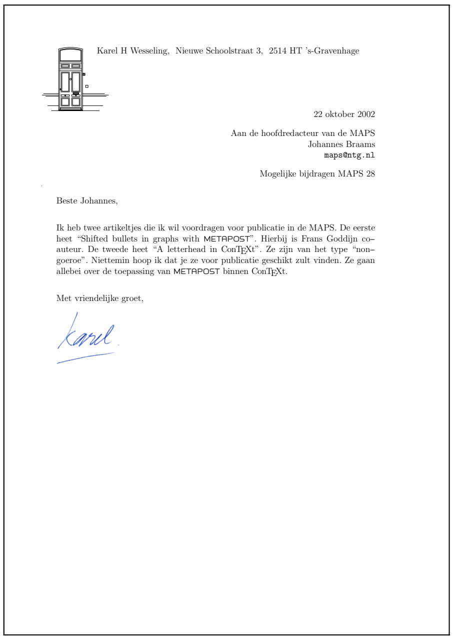
Template listing

For the convenience of the reader I provide the complete listing of the letter template.

```
% output=pdftex
\setupwhitespace[big] % Separation between paragraphs
\useexternalfigure[karelsign][kh][width=.2\textwidth]
\noheaderandfooterlines
\setupbodyfont[12pt]

\starttext
\definelayer[logohead][position=no]
\setupbackgrounds[page][background=logohead]
\setlayer[logohead][x=1.8cm,y=2cm]{%
  \externalfigure[mpdeur.pdf][scale=220]
} % end of first \setlayer
\setlayer[logohead][x=4.4cm,y=2cm]{%
  \doifmode{karel}
    {Karel~H~Wesseling,~~Nieuwe
      Schoolstraat~3,~~2514~HT~Den~Haag}
  \doifmode{hanny}
    {J~W~M~Wesseling||Gommers,~~Nieuwe
      Schoolstraat~3,~~2514~HT~Den~Haag}
} % end of second \setlayer
\language[nl]

\blank[small,force]
\rightaligned{\currentdate[day,month,year]}
\blank[medium]
% Hier de adressering % The address
%--------------------
\rightaligned{}
```

```
\rightaligned{}
\rightaligned{}
\blank[medium]
% Hier het onderwerp % The subject
%-------------------
\rightaligned{}

\blank[2*big]
% Hier de aanhef % Dear
%---------------

\blank[2*big]
% Hier de tekst % The body
%--------------

\blank[2*big]
% Hier de groet % Sincerely
%--------------

\blank[2*big]
% Hier de ondertekening % The signature
%----------------------
\doifmode{karel}{\externalfigure[karelsign]}
\stoptext
```

An example letter used to offer this manuscript to the editor–in–chief of the MAPS follows:



Karel H Wesseling, Nieuwe Schoolstraat 3, 2514 HT 's-Gravenhage

22 oktober 2002

Aan de hoofdredacteur van de MAPS
Johannes Braams
maps@ntg.nl

Mogelijke bijdragen MAPS 28

Beste Johannes,

Ik heb twee artikeltjes die ik wil voordragen voor publicatie in de MAPS. De eerste heet "Shifted bullets in graphs with METAPOST". Hierbij is Frans Goddijn co–auteur. De tweede heet "A letterhead in ConTEXt". Ze zijn van het type "non–goeroe". Niettemin hoop ik dat je ze voor publicatie geschikt zult vinden. Ze gaan allebei over de toepassing van METAPOST binnen ConTEXt.

Met vriendelijke groet,

Karel

Note in the listing that only a small selection (13) of the many available CONTEXT instructions is used. Counting a `start/stop` pair as one instruction we get:

```
\blank \definelayer \doifmode \externalfigure \language
\noheaderandfooterlines \rightaligned \setlayer \setupbackgrounds
\setupbodyfont \setupwhitespace \starttext
\useexternalfigure
```

Still, a lot is accomplished. Not that it is always easy to find the right instruction for a task. In particular when coming from LATEX (as I did) it is hard to suppress typing `\underline` and type `\underbar` instead, or `\begin{itemize}` and type `\startitemize`. On the other hand, instructions are often easy to find by searching the documentation [2] for "underline" or "itemize". This has the side benefit that one then learns of `\underbars` which underlines word for word and breaks between words at the end of a line.

Acknowledgement

I would like to thank Wybo Dekker, Frans Goddijn, and Hans Hagen for critical reading of the manuscript.

References

1. Hobby J D: Drawing graphs with MetaPost.
`\texmf\doc\metapost\base\mpgraph.pdf`
2. Hagen H: Context, the manual.
`cont-eni.pdf`
3. Hagen H: Metafun.
`metafun-s.pdf`
4. Otten T, Hagen H: Context, an excursion.
`ms-cb-en.pdf`

For the CONTEXT and MetaFun manuals I chose the screen/interactive versions in English. A4 paper printable and Dutch versions are also available, usually in the same subdirectory.

Some CONTEXT and MetaFun manuals and the CONTEXT software can be found on the TEXLive CDROM, Volume 1, now in its 7th edition. Find the subdirectory `\texmf\doc\context\`. They are also available from `http://www.pragma-ade.com/`, a CTAN site, or from the site of the Dutch–speaking TEX Usersgroup, NTG, at `www.ntg.nl`.

Fabrice Popineau
SUPELEC –
Campus de Metz
2, rue E. Belin
F–57070 Metz
Fabrice.Popineau@supelec.fr

metapost

Practical METAPOST[1]

abstract
In this article, I will explain how to pratically use METAPOST. This program is very different from usual drawing programs, but it fits very well in a TEX based typesetting system.

résumé
Ces quelques pages ont pour objectif de montrer comment utiliser pratiquement METAPOST. Ce programme est très différent des logiciels classique de dessin, mais s'intègre extrêmement bien dans la chaîne de composition basée sur TEX.

samenvatting
De volgende pagina's laten zien hoe METAPOST praktisch kan worden gebruikt. Dit programma verschilt sterk van de klassieke tekenprogramma's, maar laat zich voortreffelijk combineren met op TEX gebaseerde tekstverwerkingssystemen.

Where can METAPOST be found?

Today, METAPOST can be found in all "complete" TEX distributions:

- TETEX for Unix,
- TEXLIVE CD-ROM for Unix and Windows,
- MIKTEX for Windows,
- OZTEX, CMACTEX and if you want to use CONTEXT under OS/X then you need TETEX.

In the following treatment, I base myself upon TEXLIVE 6 that runs under both Unix and Windows. The example graphics are taken from the METAFUN Manual by Hans Hagen (see further down).
The main program (`mpost` or `mpost.exe`) is accompanied by some auxiliary programs:

- `makempx`: a control program that extracts the text parts of your `.mp` files and converts the lower level commands to a file with `.mpx` extension. This extraction is only done when the `.mpx` file is older than the `.mp` file. Once extraction is accomplished the following two programs take care of the conversion.
- `mpto`: this program extracts the `btex ... etex` parts and the `verbatimtex ... etex` parts from your `.mp` file and places them in a TEX file.
- `dvitomp`: converts `.dvi` files to `.mpx` files.

Further programs in a basic distribution include a set of so–called "macro" files to construct diagrams and graphics based on circles and boxes (for an example see figure 1). Within the TEX Directory Structure – TDS – look for these files under `/texmf/metapost`, for documentation under `/texmf/doc/metapost`, and for many examples under `/texmf/doc/guides/metapost-examples`.

The complete set of sources, documentation, examples and further contributions can be downloaded from the Comprehensive TEX Archive Network – CTAN – to be found on the Web under `ftp://ftp.dante.de/pub/tex/graphics/metapost/`. A recent

[1] First published in Cahiers GUTenberg n^{o} 41 — Novembre 2001. Authorized translation from the French by Karel and Hanny Wesseling.

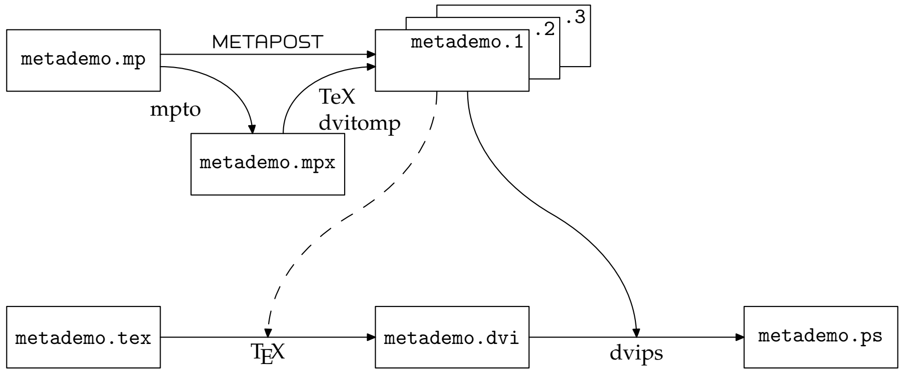


Figure 1 Compilation process of a METAPOST file

CD-ROM copy of this archive, published by Lehmanns Fachbuchhandlung is regularly distributed to members of TEX users groups without charge, but can be ordered at `bestellung@lehmanns.de`

The basics of using METAPOST with LATEX

Let us start with the two needed source documents:

`metademo.tex` containing the LATEX document, and
`metademo.mp` with the METAPOST graphics composition.

The general structure of an example METAPOST file looks like:

```
prologues:=2 ;
color c[];
c1:=red; c2:=green+red; c3:=green; c4:=blue;

def star (expr size, n, pos, color) =
  for a=0 step 360/n until 360 :
    draw (origin -- (size/2,0))
         rotatedaround (origin,a)
         shifted pos withcolor color ;
  endfor ;
enddef ;

for n = 1 upto 4:
  beginfig(n) ;
    pickup pencircle scaled 2mm ; star(2cm,n+n+3,origin,c[n]) ;
  endfig ;
endfor ;

end
```

Compilation of this file with `mpost` produces a series of four POSTSCRIPT output files, each containing one of the four declared figures (1) through (4) as shown below:

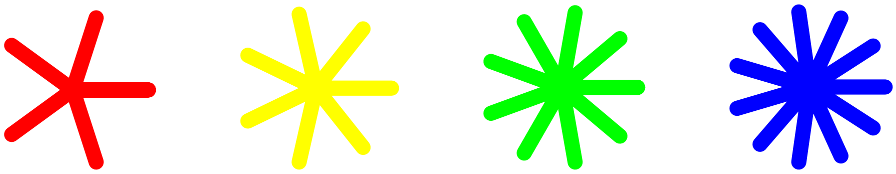
```
c:\>mpost metademo.mp
This is MetaPost, Version 0.641 (Web2c 7.3.3.1)
(metademo.mp [1] [2] [3] [4] )
4 output files written: metademo.1 .. metademo.4
Transcript written on metademo.log.
```

It is now time to include the figures just generated in your `.tex` document. This is simple when you are using a recent LATEX version in which the `graphicx` package treats the files output by METAPOST as if they were POSTSCRIPT.

For older versions you might consider renaming the files such that the file extension becomes `.eps`. For example, rename `metademo.1` to `metadem1.eps`. These files can then be handled without problems. Perhaps a better solution is to add a line `\DeclareGraphicsRule` with parameters as shown in the LATEX document below. This line specifies to LATEX that files included with the `\includegraphics` command that have an extension that is not recognized should be treated as POSTSCRIPT files.

```
\documentclass[a4paper,11pt]{article}
\usepackage{graphicx}

\begin{document}

Some ``stars'' taken from the MetaFun manual:\\[4mm]

\includegraphics[width=1.5cm]{metademo.1} \hfill
\includegraphics[width=1.5cm]{metademo.2} \hfill
\includegraphics[width=1.5cm]{metademo.3} \hfill
\includegraphics[width=1.5cm]{metademo.4}

\end{document}
```

Now compile this file and pass it to `dvips` to obtain a POSTSCRIPT file which looks like the starry result below.



In its default configuration, METAPOST does not create encapsulated, i.e. self contained, POSTSCRIPT but as long as you compile to DVI and use `dvips` exclusively to obtain a POSTSCRIPT output file, there is no problem. But if you want to use the POSTSCRIPT generated by METAPOST with other applications such as Adobe Illustrator for example, you must include in your `.mp` file a `prologue` variable which is given the value 2, as we saw in the METAPOST program above, as follows:

```
prologues := 2 ;
```

This variable is 0 by default. A positive value forces METAPOST to generate encapsulated POSTSCRIPT. Use a value of 1 for `troff`, 2 for `dvips`. With a prologue value of 2, METAPOST will look for the file `psfonts.map`, the font map file that is usually read by `dvips`. These mappings, however, are not included in the METAPOST output file. Thus, they have to be supplied together with the output, unless they already are resident in the target application. The immediate drawback of this approach is that you face problems as soon as non–ASCII character codes are used. The code vectors assumed by `dvips` will simply be ignored by other applications.

METAPOST and PDF

METAPOST maintains a particular relationship to PDF, even though it generates code that could be a priori incompatible with PDF. Fortunately, the range of instructions that METAPOST uses is a sufficiently small subset of POSTSCRIPT that it is more or less “accidentally” compatible with the PDF operators.

The CONTEXT system[2] contains a module `supp-pdf` that permits the direct conversion by TEX of EPS files produced by METAPOST into PDF instructions. The advantage of this module is that it is independent of CONTEXT and can be used in other environments, for example LATEX. When used in conjunction with PDFTEX, the `graphicx` package recognizes `.mps` files containing METAPOST instructions and charges `supp-pdf` with the conversion of such graphics.
However, METAPOST produces graphic output files with extensions `.1`, `.2` and so on, so we have to tell the `graphicx` package that such files are actually `.mps`files. Therefore we always add an IF statement as follows:

```
\usepackage{ifpdf}
...
\ifpdf
  \DeclareGraphicsRule{*}{mps}{*}{*}
\else
%  Recent LaTeX versions don't require the next line
%  \DeclareGraphicsRule{*}{eps}{*}{*}
\fi
```

Thus we see that METAPOST vector graphics files *are compatible with* POSTSCRIPT *as well as* PDF *output*. The same source files serve two output formats, and there is no need to keep and maintain two file versions, one for each output type.

CONTEXT has an equally easy utility to achieve this called `mptopdf`. It is a Perl script accompanied by a reduced format file `mptopdf.efmt` that automatically includes the necessary commands when compiling METAPOST files through `texexec`, the CONTEXT work bench of many uses.

CONTEXT also offers a new format for METAPOST called MetaFun which loads its proprietary macros on top of those of METAPOST. They include flashy effects such as the one demonstrated below (figure 2): a graded background to a line of text, going from red to yellow and back to red again.

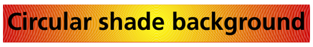


Figure 2 MetaFun's shady background.

Integration of METAPOST graphics in your source document

With `\write18`, which is part of the Web2C system, it has become possible, finally, to include METAPOST graphics transparently in your TEX document, i.e. that graphics compilation is automatically launched by TEX at the time of the compilation of the main document. But note: for security reasons the `\write18` command is disabled by default. To make it operational you must call TEX with the option `-shell_escape`, or you could modify your `texmf.cnf` file. Look for:

```
% Enable system commands via \write18{...}?
shell_escape = t
```

CONTEXT is the first provider of an environment to fully exploit the possibilities of METAPOST graphics integration in the body of a source document. See how it is done in practice by reading the code below which produces the effect shown in figure 2 above:

[2] CONTEXT, whose principal author is Hans Hagen, is a complete text composition system more modern and more ambitious than LATEX. For more information on CONTEXT look at the NTG Web site `www.ntg.nl/context/`

```
\definecolor[a][yellow]
\definecolor[c][darkred]

\startuniqueMPgraphic{CircularShade}
  path p ;
  p := unitsquare xscaled \overlaywidth
                  yscaled \overlayheight ;
  circular_shade(p,0,\MPcolor{a},\MPcolor{c}) ;
 \stopuniqueMPgraphic

\defineoverlay[circular shade]
              [\uniqueMPgraphic{CircularShade}]
\framed[background=circular shade,frame=off]
       {\bf Circular shade background}
```

A powerful aspect of CONTEXT is that it offers extensive possibilities of information exchange between TEX and METAPOST since TEX may pass parameters to METAPOST because its macros are expanded in time before the METAPOST file is generated. And using its driver modules, CONTEXT can create graphics perfectly encapsulated in typeset text composed by TEX or vice versa, as shown above with the shading example. This requires, of course, several compilation passes: the first generates the graphic definition file, the second composes the ensemble of text and graphics; between the runs metapost is called. When `write18` is enabled, only one pass is needed.

This pleasant property of CONTEXT of the fusion of figures and text in one single document has also been ported lately to LATEX thanks to the `emp` package which stands for *Embedded MetaPost*. This package can be used to produce effects similar to the example above, as demonstrated with the listing below, and the result shown in figure 3:

```
\documentclass[a4paper,11pt]{article}
\usepackage{palatino}
\usepackage{mflogo,graphicx,emp,ifpdf}

\ifpdf
  \DeclareGraphicsRule{*}{mps}{*}{}
\fi

\begin{document}

% Commands included in the Metapost file:
\empaddtoTeX{%
  \usepackage{palatino}
}

% Start of metapost figures
\begin{empfile}
  % General definitions:
  \begin{empcmds}
    color yellow  ; yellow := red + green ;
    \input mp-spec ; % or \input metafun ;
  \end{empcmds}
  % First Metapost figure:
  \begin{empdef}[fig1](5cm,5cm)
    draw fullsquare            withcolor .625red ;
    draw fullsquare rotated 45 withcolor .625red ;
    picture cp ; cp := currentpicture ;
    def copy = addto currentpicture also cp enddef ;
    copy scaled .9 withcolor .625white ;
    copy scaled .7 withcolor .625yellow ;
    copy scaled .6 withcolor .625white ;
    copy scaled .4 withcolor .625red ;
    copy scaled .3 withcolor .625white ;
    fill fullcircle scaled .2 withcolor .625yellow ;
    currentpicture := currentpicture scaled 50 ;
  \end{empdef}
  % (more figure can follow)
\end{empfile}
% On-the-fly metapost compilation:
\immediate\write18{mpost -tex=latex \jobname}

Example of a \MP\ figure included in the body of a \LaTeX\ document.
\begin{figure}[ht]
  \begin{center}
    \empuse{fig1}
    \caption{A \MP{} graphic embedded in a \LaTeX\ document}
  \end{center}
\end{figure}

\end{document}
```

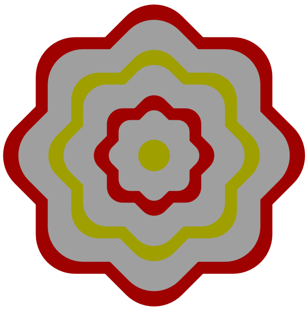


Figure 3 A METAPOST graphic embedded in a LATEX document.

Unfortunately, and unlike in CONTEXT, the `emp` package does not allow the exchange of parameters between TEX and METAPOST.

Conclusion

I hope that with this publication I have eliminated most of the practical difficulties that one may meet initially when attempting to use METAPOST. I also hope to have induced you to undergo the joys of geometry graphics. Bonne chance. Good luck.