

MAPS

NUMMER 29 • NAJAAR 2003

REDAKTIE

Wybo Dekker, waarnemend hoofdredacteur

Frans Goddijn

Taco Hoekwater

Siep Kroonenberg

Piet van Oostrum



NEDERLANDSTALIGE T_EX GEBRUIKERSGROEP



Voorzitter

Hans Hagen
pragma@wxs.nl

Secretaris

Willi Egger
w.egger@boede.nl

Penningmeester

Wybo Dekker
wybo@servalys.nl

Bestuursleden

Erik Frambach
erik.frambach@planet.nl

Frans Goddijn
frans@goddijn.com

Karel Wesseling
k.h.wesseling@planet.nl

Postadres

Nederlandstalige T_EX Gebruikersgroep
Maastraat 2
5836 BB Sambeek

Postgiro

1306238
t.n.v. NTG, Deil
SWIFT/BIC-code: ING BNL 2A
IBAN-code: NL05PSTB0001306238

E-mail bestuur

ntg@ntg.nl

E-mail MAPS redactie

maps@ntg.nl

WWW

www.ntg.nl

Copyright © 2003 NTG

De **Nederlandstalige T_EX Gebruikersgroep (NTG)** is een vereniging die tot doel heeft de kennis en het gebruik van T_EX te bevorderen. De NTG fungeert als een forum voor nieuwe ontwikkelingen met betrekking tot computergebaseerde document-opmaak in het algemeen en de ontwikkeling van 'T_EX 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 met betrekking tot T_EX en daarmee verwante programmatuur.

De NTG biedt haar leden ondermeer:

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

Lid worden kan door overmaking van de verschuldigde contributie naar de NTG-giro (zie links); vermeld IBAN- zowel als SWIFT/BIC-code en selecteer shared cost. 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 ConT_EXt formaat. Bijdragen op alle niveaus van expertise zijn welkom.

Productie. De Maps wordt gezet met behulp van een LaTeX class file en een ConT_EXt module. Het PDF bestand voor de drukker wordt aangemaakt met behulp van pdftex 1.11b Web2C 7.5.2 draaiend onder Linux 2.4. De gebruikte fonts zijn Adobe Times-Roman met Old Style Figures, Adobe Frutiger en de vrije txfonts fontset.

T_EX is een door professor Donald E. Knuth ontwikkelde 'opmaaktaal' voor het letterzetten van documenten, een documentopmaakstelsel. Met T_EX is het mogelijk om kwalitatief hoogstaand drukwerk te vervaardigen. Het is eveneens zeer geschikt voor formules in wiskundige teksten.

Er is een aantal op T_EX gebaseerde producten, waarmee ook de logische structuur van een document beschreven kan worden, met behoud van de letterzetmogelijkheden van T_EX. Voorbeelden zijn LaTeX van Leslie Lamport, $\mathcal{A}\mathcal{M}\mathcal{S}$ -T_EX van Michael Spivak, en ConT_EXt van Hans Hagen.

Inhoudsopgave

Nummer 29, najaar 2003

Redactioneel, <i>Wybo Dekker</i>	1
32 ^e NTG-bijeenkomst, <i>Frans Goddijn</i>	2
T _E X user groups worldwide – what’s cooking?, <i>Erik Frambach</i>	6
LaT _E X: een newbie-ervaring, <i>Koen Wybo</i>	10
Bach _O T _E X2003, <i>Kees van der Laan</i>	15
Toolbox, <i>Wybo Dekker</i>	24
Docbook In ConT _E Xt, <i>Simon Pepping</i>	26
Drawing Message Sequence Charts with LaT _E X, <i>Sjouke Mauw and Victor Bos</i>	38
Labels voor gevaarlijke stoffen met LaT _E X, <i>Roland Smith</i>	44
Aligning Metapost graphs in ConT _E Xt combinations, <i>Karel H Wesseling</i>	50
Drawing a type-case in ConT _E Xt, <i>Willi Egger</i>	53
Optisch uitvullen in de Maps, <i>Siep Kroonenberg</i>	60
Installing fonts in LaT _E X: a user’s experience, <i>Ferdy Hanssen</i>	61
The Font Installation Guide, <i>Philipp Lehman</i>	65

Redactioneel

Wybo Dekker
wybo@servalys.nl

Voor u ligt het negenentwintigste nummer van de MAPS. Het is het eerste nummer van dit jaar en het zal ook het enige nummer blijven. De oorzaak hiervan is dat binnen de redactie en met name bij de hoofdredacteur, Johannes Braams, te weinig tijd beschikbaar was om de leden, u dus, achter de vordden te zitten en u te bewegen tot het opschrijven van een verhaal over een van de vele leuke en voor anderen interessante dingen die u met T_EX doet. Johannes Braams heeft daarom besloten zijn functie terug te geven en het bestuur heeft Niels Moes bereid gevonden om zijn taak over te nemen. Niels is echter nog druk bezig met het schrijven van zijn proefschrift en zal daarom pas over enkele maanden zijn functie kunnen gaan uitoefenen. Hij zal de hoofdredacteur voor de voorjaars-MAPS van 2004 worden.

Tegenover die niet verschenen MAPS staat, dat we u dit jaar nog zullen verblijden de T_EXlive distributie 2003 (die deze keer ook de CTAN-dump bevat en uit twee cdroms en een dvd bestaat) en met een prachtig handzaam boekje van de hand van Hans Hagen en Willi Egger, dat een overzicht geeft van alle outline fonts die op de nieuwe T_EXlive distributie beschikbaar komen.

Verder heeft het bestuur Ernst van der Storm bereid gevonden de MAPS-redactie te versterken. De redactie zal dus bestaan uit Niels Moes, Piet van Oostrum en Ernst van der Storm. Daarnaast zullen Frans Goddijn en ikzelf vanuit het bestuur de redactie voorlopig blijven ondersteunen.

Dat wil niet zeggen dat Siep Kroonenberg en Taco Hoekwater zich niet meer met de MAPS zullen bemoeien: integendeel, zij zullen het leeuwendeel van het werk blijven verrichten. Zij vormen het productie-team, dat zich primair richt op de vormgeving en het zetten van de MAPS.

Bij gebrek aan een actieve redactie is de voor u liggende MAPS grotendeels door Frans Goddijn en mijzelf bijeengebracht en we denken dat we daar, in intensieve samenwerking met Siep Kroonenberg, in geslaagd zijn; en dat is vooral daaraan te danken dat we enkelen van u bereid hebben gevonden het hart uit te storten:

- Frans Goddijn levert een eloquent en rijk geïllustreerd verslag van de 32^e NTG-bijeenkomst.
- Erik Frambach laat zien dat ook andere Local Users Groups niet ontkomen aan problemen bij het produceren van hun tijdschriften en dat niet alleen...
- Koen Wybo beschrijft zijn ervaringen als beginneling op LaT_EX-gebied en geeft een overzicht van veel documentatie-bronnen,
- Kees van der Laan doet uitgebreid verslag van de BachoT_EX2003.
- Wybo Dekker probeert de Toolbox weer nieuw leven in te blazen.
- Simon Pepping laat zien hoe conT_EXt kan worden gebruikt om DocBook documenten te zetten,
- van twee niet-NTG-leden, Sjouke Mauw en Victor Bos hebben we een interessant verhaal uit TUGboat mogen overnemen over het tekenen van *Message Sequence Charts*.
- Roland Smith laat zien hoe hij etiketten die we in doe-het-zelf zaken en drogisterijen nog wel eens tegenkomen in LaT_EX genereert.
- Karel Wesseling laat zien hoe METAPOST-figuren onderling kunnen worden uitgelijnd,
- Willi Egger gunt ons een blik in een ouderwetse zetkast en laat en passant zien hoe de `\btable ... \etable` omgeving gebruikt kan worden om een lade daaruit te tekenen.
- Siep Kroonenberg heeft een primeur met de toepassing van pdfT_EX's protruding characters in onze MAPS en schrijft er een prijsvraag zonder prijzen over uit.
- Ferdy Hanssen neemt ons mee op de queeste van enkele font-istallaties. Hij verwijst daarbij naar Philipp Lehman's *Font Installation Guide*:
- en last, but not least: we willen meer aandacht gaan schenken aan fonts. Daarom is in dit nummer een facsimile opgenomen van de buitengewoon heldere en praktisch bruikbare *Font Installation Guide* van Philipp Lehman.

We wensen u veel leesplezier!

community

32^e NTG-bijeenkomst

Voor de 32^e NTG-bijeenkomst was “vorm en inhoud” als thema vastgesteld. Een thema waarvan je denkt “dat we daar niet eerder op zijn gekomen.” Tal van onderwerpen passen immers in die jas.

We waren voor de derde maal te gast in het Arnhemse Hotel Haarhuis. Na de koffie in de bekende wandelgangen van het hotel begonnen we de dag met een ledenvergadering. Van het bestuur zaten voorzitter Hans Hagen en secretaris Willi Egger aan de tafel voorin de zaal. Eerst dreigde de hele agenda (notulen, mededelingen, ingekomen stukken...) in vijf minuten gedaan te zijn, inclusief vrolijke mededelingen over de maps (komt eraan) en 4 \TeX (conflict opgelost, Wietse Dol vriendelijk akkoord, Erik laat zijn aandeel in de opbrengst aan de NTG, applaus).

Maar gelukkig kwam het onderwerp OSOSS ter sprake, een ingewikkelde kwestie waar slechts enkele experts het woord over kunnen voeren. Dat deden deze deskundigen dan ook en wel zodanig dat de geplande vergadertijd ruim werd gevuld.

Piet van Oostrum opende de reeks voordrachten met een volledige verhandeling over bib \TeX , een onderdeel van \TeX waarmee bibliografieën zijn te maken bij opstellen, artikelen en boeken. Piet had de voorafgaande nacht tot twee uur doorgewerkt aan zijn presentatie en zoals we van hem gewend zijn, leverde hij een perfect verhaal. Naast de technische aspecten was het ook interessant om te horen hoe merkwaardig men in de ons omringende landen denkt over alfabetisch sorteren. Bij het op de gewenste volgorde krijgen van een lijst van auteursnamen leidt dit tot allerlei verrassende ingrepen.



Piet van Oostrum,
door Dilys Bronkhorst



Ledenvergadering geleid door voorzitter Hans Hagen (r)



Johan Vromans vertelt over MMDS.

Intussen was gasttekenares Dilys Bronkhorst gearriveerd en zij zou gedurende de dag een aantal portrettekeningen maken.

Johan Vromans schaamt zich er niet voor een hacker te zijn. Hij is er juist trots op. Hij vertelde over het ontstaan van MMDS, zijn op Perl gebaseerde pakket waarmee mensen die echt niets van $\text{T}_{\text{E}}\text{X}$ weten toch door middel van $\text{T}_{\text{E}}\text{X}$ documenten kunnen maken. Johans Perl-scripts behappen de door de gebruiker ingetikte tekst, begrijpen wat een aanhef is, wat een kopje, een opsomming... en de brief, het artikel en de offerte worden keurig in $\text{T}_{\text{E}}\text{X}$ gezet en voorzien van briefhoofd en paginering drukklaar gemaakt. Overigens, ook webpagina's kunnen op dezelfde manier worden vervaardigd. De ontwikkeling van MMDS is tevens een afspiegeling van de computerhistorie, want Johan Vromans werkt al aan het pakket sinds de jaren tachtig van de vorige eeuw.

Tijdens de lunch konden de NTG-leden onderling ideeën uitwisselen en aan alle tafels klonk een enthousiast rumoer. Dilys Bronkhorst tekende er een interpretatie van René van der Heijden.

Sanne Dijkstra was vlak voor de lunch gekomen, niet alleen omdat we er bekend om staan onze gasten in de watten te leggen maar ook om vlak voor haar lezing even te peilen wat voor volkje $\text{T}_{\text{E}}\text{X}$ gebruikers nu eigenlijk vormen. Er was haar verteld dat het gaat om nerds met verstand van typografie, maar met een andere bril op dan normale typografen. Haar voordracht was voor velen in de zaal een fris en ander geluid: typografen op een kunstacademie kijken anders dan wij gewend zijn naar de wetten en regels van het zetwerk. Sommige dingen die Sanne in haar verhaal aanstipte zijn voor ons gesneden koek, zoals de onwenselijkheid van kleinkapitalen als deze worden verkregen door hoofdletters domweg verkleind te zetten, of het vet maken van letters door bestaande letters in dikker zwart te plaatsen. Andere benaderingswijzen van vlakverdeling waren voor een aantal luisteraars wel nieuw, zoals het bepalen van een ideaal vlak voor tekst op een vel door middel van het tekenen van diagonalen over een samenstel van twee pagina's. Tenslotte gaf Sanne aan de hand van dia's van de $4\text{T}_{\text{E}}\text{X}$ manual haar suggesties voor de manier waarop je kritisch naar functionele vormgeving zou kunnen kijken.



Johan Vromans,
door Dilys Bronkhorst



Sanne Dijkstra



Tenar van Kooten Niekerk en pianiste Cathelijne Maat



René van der Heijden,
door Dilys Bronkhorst

Wonderlijk mooi sloot de voordracht van Willi Egger hier op aan. De diagonalen die door Sanne Dijkstra werden genoemd, bleken een startpunt van zijn verhandeling over gulden snede en eenvoudige constructietekeningen waarmee feilloos kan worden bepaald waar je het beste je kantlijnen kunt plaatsen en hoeveel tekst je volgens beproefde vuistregels op je pagina kunt zetten. Daarbij deed Willi ook uit de doeken hoe het komt dat boekjes die op A5-formaat zijn geproduceerd meestal zo vervelend in de hand liggen. De afgelopen keren dat Willi Egger voordrachten hield op NTG-dagen bleek de tijd telkens zo krap bemeten door uitgelopen lezingen dat hem was verzocht zijn verhaal wat in te korten, reden waarom we ditmaal nadrukkelijk een héle lezing hadden gevraagd. Niemand had daar spijt van en na het verhaal van Willi kwamen er allerlei vragen en discussies los, beantwoord en gestuurd door zowel Willi Egger als Sanne Dijkstra, die speciaal voor de lezing van Willi was gebleven.

Ferdy Hanssen liet daarna zien hoe je, door de aanwijzingen van een paar handleidingen secuur en geduldig toe te passen, zonder al te veel vallen en opstaan zelf een Type1 font kunt installeren. Hoe ingewikkeld dat is, hangt deels af van de keuze van het font. Voor veel fonts staat al bijna alles op de $\text{T}_{\text{E}}\text{X}$ Live CD/DVD, voor andere moeten tal van bestanden worden gegenereerd, gekopieerd, veranderd van naam...

Na de theepauze nam Hans Hagen plaats achter de sprekerstafel. Als Hans vertelt over de nieuwste ontwikkelingen dan is het opletten geblazen. Soms verschijnen er dermate nooit-vertoonde beelden op het beamerscherf dat er enige tijd overheen gaat voordat je je realiseert wat er eigenlijk aan de hand is. Als je dan een deel van Hans' tekst hebt gemist, moet je je stevig concentreren om de boot niet ernstig te missen. In dit geval kwam er een PDF document op het scherm waar allerlei dingen op konden worden ingevuld. Andere schermen werden geopend en gesloten en een typisch $\text{T}_{\text{E}}\text{X}$ -run log-scherf flitste voorbij waarna een ander document in PDF werd bekeken.

Na verloop van tijd bleek het een systeem te zijn dat in principe verwantschap heeft met de aanpak van Johan Vromans' MMDS. Hier kon de gebruiker allerlei keuzes en opties menugestuurd aanklikken, waarmee op simpele wijze mogelijkheden beschikbaar komen die er normaal alleen zijn voor mensen die niet bang zijn van gecompliceerde command line variabelen. Een opvolger voor $4\text{T}_{\text{E}}\text{X}$?



Michael Guravage smeedt $\text{T}_{\text{E}}\text{X}$ plannen met Patrick Gundlach.



Roland Kwee (l) en Willi Egger (r)

Hans heeft weleens verteld dat bij grotere instellingen die T_EX voor publicaties gebruiken de trend bestaat om T_EX te beschouwen en te gebruiken als back-end, als een applicatie waar de meeste gebruikers niet veel meer mee te maken hebben omdat T_EX op de achtergrond zijn werk doet. Deze nieuw getoonde schil, een slim PDF document waarmee op de achtergrond allerlei T_EX trucs kunnen worden aangestuurd, lijkt ook die richting op te gaan.

Intussen arriveerden zangeres Tenar van Kooten Niekerk en pianiste Cathelijne Maat. Ondanks de erbarmelijk ontstemde piano van het hotel brachten de beide vrouwen met grote flair en helderheid een sprankelend programma. Deze weken staan zij met twee collega's in een Arnhemse theater met een show rond een reeks liedjes over de liefde. Willi Egger had een aantal liedteksten met behulp van ConT_EXt in een kleine programmafolder gezet.

Mede dankzij het inspirerende en verfrissende optreden van Tenar en Cathelijne waren de leden vervolgens in een puike stemming om tijdens de borrel in het hotelcafé over van alles met elkaar te brainstormen. Niet iedereen bleef over T_EX bezig, maar hier en daar werden toch nog T_EX-plannen gesmeed.

Ook tijdens de eenvoudige doch voedzame maaltijd in het restaurant van Haarhuis was het typesetten her en der nog onderwerp van gesprek, maar in het algemeen ging de conversatie meer de richting op van vrije kunsten, atletiek en liefde. Toch was er ook nog gelegenheid voor een korte vergadering aan een tafeltje apart, van de mensen die momenteel met de Maps bezig zijn.

Ja, het werd erg laat, maar er ging deze dag geen uur verloren!



Siep Kroonenberg,
door Dilys Bronkhorst



Erik Frambach geniet van
een geslaagde NTG-bijeenkomst.



Dilys Bronkhorst met Frans Goddijn

community

T_EX user groups worldwide – what’s cooking?

Erik Frambach

abstract

This article is based on a presentation given at the UK TUG meeting in Oxford in October 2002. It describes some current problems that T_EX user groups face and it attempts to distill lessons learned and recommendations from almost 25 years of T_EX user groups history.

keywords

T_EX user groups, lugs, problems, volunteer work, history

Introduction

During discussions by email and at T_EX meetings in The Netherlands, Germany, Poland, the UK and the USA, it became clear that currently several T_EX user groups (local user groups, ‘lugs’ for short) are facing similar problems. These problems include constitutional crises, failures to produce journals, membership decline and financial difficulties.

We will point out and discuss some of these problems and we will attempt to provide recommendations for solving and/or preventing them.

History

Since the ‘birth’ of T_EX in 1978 many lugs were formed. In Table 1 we’ve listed some highlights in lugs history. It’s remarkable that most lugs were formed in the last decade. Undoubtedly the Internet has played a major role here by providing easy and cheap means of software distribution (FTP servers and World Wide Web servers) and personal contacts (Usenet News and email).

Gathering a historic overview of the founding (and dissolving) of lugs proved to be rather difficult. No one seems to be keeping track of the history of lugs. Even websites of existing user groups often provide little or no information on their history.

Analysis of the development of user groups over the years shows a distinct geographical bias:

- Europe: very many lugs were founded.
- North America: very few lugs were founded.
- Asia: only a few lugs were founded.

- South America: no lugs were founded.
- Africa: no lugs were founded.
- Other continents: neither.

The many European lugs can be explained by the need for support for their own language in T_EX (e.g. DANTE, NTG, GUTenberg, CSTUG). This also (partially) explains why there are so few in North America. The oldest lug (TUG) resides there, and by nature T_EX is English language oriented. Nevertheless, the vastness of the continent could have encouraged more local groups to start their own activities. In Asia T_EX seems to be much less known than in the western world. Although typesetting eastern languages with T_EX is possible, it’s hardly as easy as typesetting western (Latin based) languages. There may also be a cultural barrier here. South America and Africa (in fact, all continents on the southern hemisphere) lack lugs. We can only speculate on the reasons for this.

Ups and downs

Many lugs have had and/or are having their share of problems, but there are also successes to celebrate. In this section we will list some of them.

Internal problems

Running a lug is not a trivial task. Many things can go wrong. Some of the problems that have occurred are:

- Constitutional problems: bylaws and articles may prove to be extremely limiting or paralysing under difficult or un-

1978:	T _E X ‘final’ version released
1980:	TUG founded
1987:	UK TUG founded
1988:	NTG founded, GUTenberg founded
1989:	DANTE e.V. founded
1992:	GUST founded
1997:	19 lugs
2002:	25 lugs

Table 1. Highlights in user groups history

foreseen circumstances. E.g., constitutions may require a certain hard coded number of members to be present at a meeting for any decision to be taken. Sometimes nothing is specified about dissolving the lug which can lead to endless discussions.

□ Boards that don’t perform (well): struggle for power, incompatible personalities, poor discussion technique, poor management and hidden agendas are only a few of the problems that a lug (or any organization) may have to face. Part of the problem is that the goals of a lug are often unclear or too abstract (“promote usage of T_EX”).

□ Financial problems: fuzzy bookkeeping, insufficient insight and control, an amateurish approach and poor (independent) auditing are some the problems that have occurred. Big risks can be involved. Who can afford a big conference like EuroT_EX? What will happen if the tax man starts an inquiry?

□ CD-rom production problems: we’ve seen 4T_EX thrive for several year, but then it died because of internal conflicts. T_EXlive has been very successful but is very vulnerable because of the extremely small size of the team that produces it.

□ Journals: MAPS, Baskerville, TUGboat, T_EXnische Komödie are having problems acquiring enough articles and/or publishing on time.

□ Meetings: the organization and timing of meetings is often more or less random, cheap locations can be hard to find, conferences can be costly (risky), and some lugs opt for electronic only “meetings” (e.g., the Nordic group).

□ Volunteers: you can’t rely on them to live up to expectations (4T_EX, T_EXlive, journals, conferences, etc.). You can’t force them either, so you need a more subtle management style to encourage and support them.

International cooperation

Although all lugs have similar goals, their activities may interfere with each other and opportunities for cooperation and synergy may be missed.

□ Conferences: lack of coordination may cause dates of major meetings to clash; too often the organization depends on the same volunteers; financial risks are unclear and certainly not covered by the whole group of lugs.

□ Journals: distribution of journals to all lugs is still not in place; exchange of interesting articles and translation services are still poor; indexes of all journals or an index of all journals combined is lacking – better still would be a database, but that is definitely beyond reach; online availability of journals/articles and crossreferencing is very poor or non-existing.

□ Finance: banking costs are not (well) managed, resulting in loss of money; conference budgets and reports on financial results are not standardized or not available at all.

□ EuroT_EX & TUG Conference: coordination and selection of organizing parties is very obscure – nevertheless conferences are usually very successful.

□ pdfT_EX: international cooperation made this product flourish; NTS: many lessons were learned but this academic exercise doesn’t justify its costs; Omega: lack of cooperation repels users and volunteers and frustrates development.

The Good News

The gloomy list of internal problems above may give you the impression that lugs are doomed. Indeed there is lots of room for improvement, but let’s not forget what we’ve achieved. Here’s an overview of only a few successes in which lugs played an important role:

□ T_EXlive cd-rom: this product made T_EX easily accessible to users on all major operating systems, standardizing the way T_EX installations are configured.

□ CTAN and CTAN cd-roms: CTAN has been a valuable repository of all T_EX related software where countless users with Internet access have found the resources they needed. The cd-rom versions are very useful for those with no or limited access to the Internet. Though not intended to, the cd-roms also serve as historic snap shots by freezing archives at regular intervals.

□ 4T_EX: this product was the first to provide a true plug & play T_EX environment that could run completely from cd-rom (remember, disk space was expensive back then) and contained all tools usually found in commercial word processors.

□ pdfT_EX: this product instantly warped T_EX into the new era of PDF documents and all their extra features like hypertext and web links. pdfT_EX even made Adobe admit it had no software that could produce documents of the level of complexity that pdfT_EX could (easily) generate.

□ NTS: this ‘New Typesetting System’ was initially supposed to be the successor of T_EX as we know it, but eventually it turned out that for various reasons it could only be a complete rewrite of T_EX in a different programming language which should make the implementation of extensions much easier. This may or may not happen, but anyway this project proved that the T_EX community at large, represented by many lugs, is capable of (re)building a system as complex as T_EX. A true successor to T_EX will become essential soon enough as the world progresses...

□ Mailing lists and news groups: several of these have been very successful in providing support to T_EX users of all levels, and as a platform for discussions of various T_EXnical issues.

□ Conferences: every year there are at least a few successful meetings which last half a day up to a week, where progress is made by discussing T_EXniques and where representatives of lugs can gather to coordinate their efforts in a friendly environment.

□ International contacts during lug meetings: over the years some of the meetings of lugs which used to be ‘local’ have become more and more international (e.g., GUST, DANTE, UKTUG and NTG meetings).

□ New: Webcalendar is a web based system that all lugs can use to publish their meetings and other events to the whole world. Potential date clashes are easy to track because the Webcalendar can show all events of all lugs simultaneously.

□ New: a central ‘T_EX bank’ account should help in minimizing international money transfers which are always costly.

□ New: an attempt has been made to get European funding for the further development of T_EX. This is a joint effort of several European lugs.

Patterns or Lessons learned

Looking back at more than two decades of lugs at work, we can distinguish some recurring patterns, and there are several lessons to be learned:

□ Amateurs operating on (semi) professional levels can deliver astonishing products. But, reliability and consistency are weak.

□ There is no shortage of money, but not enough activities to spend it on. There are no established procedures for requesting support from lugs.

□ Volunteers are tough to manage, especially by volunteers :-). It’s not clear how the precious time and energy of volunteers can be managed and/or supported in a successful way.

□ New lugs are mostly oriented on ‘difficult’ languages (e.g., Indian, Chinese, Hungarian, Vietnamese), not on application, working field or computer system.

□ Formalities have frustrated many projects and activities. However, anarchy hasn’t done much better.

□ Many successes in the T_EX world can be attributed to individuals, with little or no involvement/support of lugs

(e.g., L_AT_EX, e_mT_EX, CTAN, Web2c, ConT_EXt, Omega, f_pT_EX, MikT_EX).

□ We don’t learn from each other’s mistakes. There is no sense of history. All too often a new lug board means that much knowledge and experience goes down the drain.

Do’s and Don’ts

From the experience gained by lugs we can extract the following recommendations to lugs:

□ Review your constitution (compare it to others) on a regular basis. Watch out for potential deadlocks (voting rules!) and unforeseen events such as dissolution of the lug.

□ Check your legal and financial position on a regular basis. Get independent professional advice before it’s too late (read: the tax man will strip you down...).

□ Find the right size of your board (not too big, not too small) so you can actually get things done.

□ Think twice before you hire personnel e.g. for running a lug office. This could generate more problems than it will solve.

□ In case of internal problems, talk to other lugs. Chances are your problems are not unique.

□ Cherish your volunteers! They are your most valuable assets.

Bold statements

Naturally we don’t have all the answers to all the problems. However, we do have several items we think would be useful for lugs to discuss among themselves or with other lugs. We list them here as ‘bold statements’. Note that we deliberately chose provocative expressions and that they do not necessarily represent our view on the issue. They are primarily meant to evoke discussions.

1. There is no good definition of a lug. There are no criteria for becoming a ‘formal’ lug. Or an ‘informal’ lug, whatever that is.

2. Vision and mission of lugs are poorly developed. Most, if not all, lugs have no long term strategy, which makes them vulnerable to lots of problems.

3. The model of user groups based on a common country or language no longer suffices.

4. Information and expertise is no longer hard to find because of the Internet. Therefore lugs have to find new reasons for being. A new lug model is needed.

5. Most lugs are clueless when it comes to promoting T_EX or even themselves. This is a threat for the entire T_EX community.
6. Lugs should be more involved in the process of defining and implementing a successor to T_EX and accompanying software.
7. Inter-lug communication depends only on a few persons who meet only a few times a year. Email communication among lugs is subminimal. Joint projects could help to improve this.
8. Intra-lug communication should be improved. Sending a cd-rom and a journal, and organizing one or two meetings a year is not enough. Members will not become involved, which is why it’s hard to find volunteers.
9. Lugs should not have professional staff members. This scenario is bound to lead to problems, as history has proven.
10. Lugs growing too big to handle (2000 members?) should split into smaller groups.

Conclusion

For already two decades lugs have been around to promote T_EX and support T_EX users. They have had their ups and downs, but most of them have survived in spite of the incredible advances and changes that the computer world has gone through and is still going through. We are sure that in the next decade lugs should play a major role in keeping T_EX and the whole T_EX community alive and kicking. So let’s redefine and revive lug activities!

advocacy

LaTeX: een newbie-ervaring

abstract

Hoe ik een LaTeX-adept werd; argumentatie pro LaTeX en contra zijn grafische concurrenten: Word en OpenOffice.

Inleiding

Reflecterend op de weg die ik heb afgelegd om uiteindelijk bij LaTeX aan te komen, kan je niet anders dan verwonderd zijn. Want na negen jaar werken met diverse varianten van grafische tekstverwerkers ben ik nu tevreden met mijn prestatie die ik maak met een simpele editor. Graag vertel ik je ‘hoe het zover is kunnen komen’ ;-), waarom ik uiteindelijk met LaTeX werk en geef ik graag nog enkele aanbevelingen voor newbies.

Kleine routebeschrijving

Het is gek om te bedenken dat ik zo’n tien jaar terug mijn thesis heb verwerkt met een elektronische typemachine met geheugen. Van opmaak was niet veel sprake. Ik beschikte slechts over twee typmagrietwielen die gebruikt werden voor respectievelijk gewone tekst en voetnoottekst. Door te werken met hoofdletters, onderstrepen en vet kon uniforme lay-out worden aangebracht. Voor mijn studentenbudget was dit wonder van techniek een prijzige aankoop. Financiële argumenten hebben toen de doorslag gegeven want de veredelde typemachine kostte toen maar de helft van een PC zonder software (jawel het tijdperk van de i286) en was zelfs drie keer goedkoper dan een Mac. Gevolg van het budgetteren is dat ik nu enkel een getypte versie heb van mijn thesis en de digitale variant enkel kan geraadpleegd worden op de typemachine die de tekst mooi presenteert op een scherm van 10 regels hoog. Conversie naar PC of Mac is niet mogelijk.

Het eerste schooljaar dat ik les gaf maakte ik op deze machine mijn — weinige — cursusmateriaal. Maar de informatica was niet langer te stuiten: een tweedehands PC gekocht met daarop Word6. Een wereld van mogelijkheden ging open. Meerdere lettertypes, invoegen van clipart, landscape afdrukken, uitwisselen van documenten ... Het nodige overtypewerk werd erbij genomen als ook de eerste strubbelingen met nukkige software.

Groot was m’n opluchting toen de opvolger Word97 verscheen. Dit pakket had duidelijk meer onder de motorkap. M’n enthousiasme leidde zelfs tot het delen van m’n opgedane kennis. In de scholengroep waar ik werkzaam ben, gaf ik meerdere navoringsessies Word97.

Het gratis pakket Staroffice 5.1 wist mij te verleiden om over te stappen. Niet enkel omdat het gratis is, maar ook omdat het een platform-onafhankelijk programma is. Staroffice was/is immers ook verkrijgbaar voor Linux, een besturingssysteem dat vanaf 1999 meer en meer mijn aandacht kreeg. Gelukkig is de dochter: OpenOffice [13] in zijn huidige toestand heel wat stabiel en sneller.

De eerste stappen met LaTeX dateren van rond de milleniumwende. Toen echter kreeg het een onvoldoende en heb ik het lang niet gebruikt. De steile leercurve was er op dat ogenblik teveel aan, de tijd ontbrak om alles op het net op te zoeken. Pas nadat er bij een samenwerkingsverband met linux-vrienden werd geopteerd om documenten aan te maken in LaTeX, heb ik het echt weten waarderen. Het begin was even doorbijten totdat de

basisprincipes geleerd waren. Nadien kreeg ik mijn werk gedaan. Maar zoals altijd wenst men meer: speciale hoofding, aangepaste paginavoetnoten, invoegen van afbeeldingen met tekstomloop, conversie naar html en pdf... De zoektocht naar meer informatie op internet en usenet was begonnen.

Waarom overstappen?

Naast de ‘sociaal-gedwongen’ overstap naar LaTeX zijn er meerdere argumenten waarom ik niet afkerig ben van LaTeX.

Emotioneel

WYSIWYG mag dan als slogan vlot verkopen, het werkt toch niet zoals beloofd. Pagina-eindes die niet meer weg te halen zijn, afbeeldingen die maar niet op de juiste plaats willen blijven staan, opmaak die verdwijnt na het knippen/plakken in een ander bestand, hoofdingen die ‘plots’ de standaardstijl niet meer respecteren, enzovoorts... Al deze ellende begint je na verloop van tijd de keel uit te hangen. Gekoppeld aan het gevoel dat jij moet voldoen aan de software in plaats van omgekeerd, leidt dit bij mij tot de conclusie dat WYSIWYG-software niet ‘optimaal’ is.

Een ander belangrijk emotioneel aspect is toch wel het esthetisch verschil tussen pagina’s opgemaakt met LaTeX en die van een grafisch tekstverwerkingsprogramma. Daar kan je alleen maar jaloers op worden.

Economisch

Je zou het nauwelijks verwachten dat ook een Vlaming naar zijn portemonnee kijkt. En toch... Schaf je je verschillende versies aan van het officepakket van de marktleider dan mag je tweejaarlijks een behoorlijk bedrag op tafel leggen. Je kan het natuurlijk wel bij het oorspronkelijke product houden van een tiental jaar terug maar dan is het hopen dat je collega’s niet upgraden naar meer recente versies. Jij zal dan onmogelijk hun document kunnen openen. Om nog maar te zwijgen van de restrictieve licenties, de Rights Management Services en de verplichte upgrade van het totale Windows besturingssysteem die de maker van het meest verspreide office-pakket zal inbouwen in zijn nieuw product.¹ Niet te vergeten natuurlijk de aanschaf van een virusscanner want deze documenten blijken soms ‘beestjes’ in zich te dragen.²

Als tijd geld is, dan is de overstap naar LaTeX zeker aan te raden. Door het gescheiden houden van inhoud en vormgeving spaart mij dit heel wat uurtjes oplapwerk uit. Niet enkel voor mijn eigen documenten maar ook deze die ik ontvang van collega’s. Immers door printerinstellingen, aanwezige lettertypes, enzovoorts... verschilt een Word-document op twee verschillende computers.

Tekstverwerkingsprogramma’s zorgen voor het nodige comfort om op een efficiënte wijze je resultaat (in mijn geval: een cursus godsdienst) te verwezenlijken. Men kijkt graag naar dit gebruiksgemak en gaat al te vaak voorbij aan de effecten van het gebonden zijn aan een bepaald programma op lange termijn. Ben je zeker dat je na twintig jaar je documenten nog kan openen? Hoeveel geld zul je in die tijd gespendeerd hebben aan ‘noodzakelijke’ upgrades? Stel dat de producent van het programma failliet gaat: beschik je over voldoende conversiemogelijkheden om dit naar een ander programma over te brengen? Al-

1. voor de nieuwe versie van Word2003 plant Microsoft een nieuwe documentindeling. Het zal voor mensen van de ‘oude’ versie onmogelijk zijn de nieuwe versie te openen en te bewerken. Ook zal de conversie van Word-documenten door andere grafische tekstverwerkingsprogramma’s zal niet meer mogelijk zijn. [17]

2. Gelukkig bestaat er voor mensen die liever grafisch werken een gratis office-pakket dat zeker niet moet onderdoen voor zijn commerciële tegenhangers. Openoffice [13] is niet enkel kostenloos, het realiseert bovendien nog één van mijn principiële doelstellingen.

lerlei zaken die je in overweging moet nemen, wil je niet gebonden zijn aan één bepaalde producent. Niet enkel de grafische tekstverwerkers lijden aan dit euvel. Ook bepaalde \TeX -varianten zijn van commerciële aard.

Economisch betekent ook dat je software gewoon moet werken. Wat ben je met een Office-pakket waar je jezelf continu moet herinneren om tijdig op te slaan omdat een crash op de minst verwachte, meest ongelegen tijdstippen kan plaatsvinden. Anders uitgedrukt: crashes genereren altijd een vorm van verlies. Zij het in verlies van typtijd of inspirerende gedachten ‘van het moment’ die verloren zijn gegaan.

Principieel

\LaTeX is ‘Vrije software’. Dit betekent dat elke gebruiker het recht heeft om de software te gebruiken, kopiëren, verspreiden, bestuderen, veranderen en verbeteren.³ Het resultaat hiervan is zichtbaar in de opkomst van het GNU/Linux-besturingssysteem: een juweeltje van OS dat naast de kernel een gans scala aan vrije software omvat. Deze GNU-gedachte brengt een ‘ecosysteem’ tot stand waar mondiaal mensen werken aan software die voor iedereen beschikbaar is. Hierdoor ontstaat niet een — wat sommigen noemen — ‘communistische’ variant van software maar brengt het betere kwaliteit tot stand. Bovendien ontstaat zo een schare aan mensen die zich willen inzetten voor het project en is vanaf een bepaald ‘point of no return’ ook de overlevingskans vergroot. Herman Bruyninckx, prof aan de K.U. Leuven en heftig voorstander van vrije software, heeft heel wat argumenten [14] in het voordeel van Open Software.⁴ Producten gebaseerd op \TeX die niet open zijn, beveel ik niet aan bij vrienden die ook met een \TeX -variant willen beginnen. Een uitspraak die erg cru lijkt. Uiteindelijk komt het neer op je niet laten inpalmen door een producent die je afhankelijk wil maken van zijn software.⁵ Voor je het weet ben je vastgeklonken aan één bepaald product. Naast een vlotte interface is het wijs om oog te hebben voor duurzaamheid op lange termijn.⁶ Het voordeel van een open standaard is bekend. Internet heeft er zijn bestaan aan te danken. Juist door het feit dat iedereen de technische aspecten van de standaard ter inzage heeft, kan het niet gemonopoliseerd worden door één bepaalde fabrikant. Het naleven van een standaard garandeert een langere levensduur van de software: standaarden worden pas vervangen als er technologische nood aan is, niet als één of ander bedrijf verwacht er commercieel voordeel uit te halen. Daarom ook is \LaTeX zalig om mee te werken. Je bent gerust dat:

1. heel veel mensen werken met de software en je dus altijd wel ergens terecht kan met je vragen
2. de software wordt uitgebreid en onderhouden

3. \LaTeX is opgenomen in de lijst van de vrije software [1]. Meer info over vrije software is te vinden op [2]

4. Je kan het eens nalezen op [15] Op [4] krijg je nog enkele argumenten mee voor efficient ICT-gebruik die ook deels ook opgaan voor de \LaTeX -gebruiker. Ook op [8] krijg je argumenten mee waarom een pakket als \LaTeX te verkiezen is boven een WYSIWYG.

5. Voor extra argumenten: zie supra, het internetartikel van Dhr. Bruyninckx

6. Naast open standaarden bestaat er inderdaad ook open content. Een thema dat we hier buiten beschouwing laten.

Richard M. Stallman, Linus Torvalds, and Donald E. Knuth engage in a discussion on whose impact on the computerized world was the greatest.

Stallman: *God told me I have programmed the best editor in the world!*

Torvalds: *Well, God told me that I have programmed the best operating system in the world!*

Knuth: *Wait, wait – I never said that.*

From rec.humor.funny. submitted by ermel@gmx.de (Erik Meltzer)

3. zonder de bedoeling om je financieel uit te knijpen als een citroen
4. en je weet dat je kennis van het pakket niet weggeërodeerd zal zijn na de volgende o-zo-noodzakelijke update.

It takes guts to LaTeX

LaTeX is niet voor iedereen. Volgens mij zijn er enkele basisvoorwaarden waaraan je moet voldoen om met LaTeX te kunnen werken. Voor alle duidelijkheid: spreek hier enkel vanuit mijn ervaring met TETEX, de UNIX-variant.

1. WYSIWYG geeft je het gevoel alsof je controle hebt over de software en je eindproduct dat je maakt. Je hebt een instant-kijk op je eindproduct. Met een paar muisklikken boks je een tekst in elkaar die voldoet aan je inhoudelijke en esthetische normen. Om de stap te zetten naar een programma dat je terugwerpt naar het inhoudelijke en het esthetische voor eigen rekening neemt, is veel moed en lef gevraagd. Als je die stap kan maken, ben je goed op weg om ook LaTeX een kans te geven.
2. Wil je een serieuze inspanning leveren om je teksten, artikels, boeken ... ook op esthetisch vlak een voltreffer te maken, dan kan LaTeX je ‘companion’ zijn.
3. Je moet durven in te gaan tegen de algemene mentaliteit van collega’s en vrienden die grafisch nog altijd gelijkstellen met superieure programma’s. Bovendien wordt een Worddocument gezien als een ‘standaard’.⁷ Het is bijna ongehoord om een alternatief te formuleren.
4. Als je de beslissende keuze maakt om te gaan voor jouw eindproduct en niet voor het programma waarmee het moet gemaakt worden, dan ben je op de goede weg.
5. Uiteindelijk: als je HTML kan schrijven in een editor: waarom dan ook niet LaTeX?

Startblok voor nieuwe benen

Nuttige tips die je kan geven als iemand aan LaTeX begint. Anders uitgedrukt: om de frustratiedrempel te verlagen is het goed om:

1. Op weg gezet te worden met iemand die reeds LaTeX kent.
2. Of je kan je natuurlijk inschrijven op de mailinglijst van de NTG.⁸ Ook in de (engels-talige) discussiegroep `comp.text.tex` is er heel wat info te vragen (en te vinden).
3. Zoek eerst documentatie op en vooral: lees ze aandachtig. Heel wat documentatie is reeds online te vinden. Naast de NTG-site [11] kan je ook je inspiratie opdoen bij [9] en [16] Een algemene beginnersinleiding op LaTeX is te vinden op [7] De elektronische versie van MAPS [10] is ook niet te versmaden.
4. Wil je na de eerste stappen heel specifiek gaan werken dan zijn de boeken van onder andere Leslie Lamport [6] en Michel Goossens [3] uitstekende begeleiders. Ook *A Guide to LATEX* [5] krijgt positieve recensies.⁹ Je kan ze kopen of — goedkoper — bij de bibliotheek ontlenen (eventueel via het bibliotheek-uitleensysteem.)
5. Weet ook dat er bij LaTeX onopgeloste mysteries blijven bestaan. De meest hardnekkige W-vragen die je zal blijven stellen zijn:

⁷. Wat het uiteraard niet is omdat de specificaties niet zijn.

Zie ook <http://www.openstandaarden.be/teksten/definitie>

⁸. Inschrijven kan op [12]

⁹. De meeste van deze boeken krijgen in het voorjaar van 2004 een update.

- Wanneer verschijnt LaTeX3?
- Waarom werken niet meer mensen met dit systeem?
- Waarom enkel EPS in LaTeX en niet JPG?
- Waar ergens op de CTAN-server?
- Waar is mijn fontmap en hoe gebruik ik nu die fonts?
- Wat is het verschil met XML en welke is nu de beste?
- Wie maakt die LaTeX-packages en waarom?
- Waar vind ik voorbeeld-PDF's?
- Welke editor/front-end is gemakkelijker dan Vi?

Referenties

- [1] <http://www.gnu.org/directory/all/>. Lijst van vrije software.
- [2] <http://www.gnu.org/home.nl.html>. Nederlandstalige gnu-site.
- [3] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX companion*. Addison-Wesley, 1994.
- [4] <http://people.mech.kuleuven.ac.be/~bruyninc/efficiente-ict/#achtergronden>. Argumentatie waarom open standaarden te verkiezen zijn boven propriëtaire in het uitwisselen van documenten.
- [5] Helmut Kopka and Patrick W. Daly. *A guide to LaTeX2ε: Document Preparation for Beginners and Advanced Users*. Addison-Wesley, third edition, 1999.
- [6] Leslie Lamport. *LaTeX: a document preparation system*. Addison-Wesley, second edition, 1994.
- [7] <http://ludit.kuleuven.be/software/latex/>. Introductiepagina LaTeX van de K.U. Leuven met heel wat interessante links voor newbies.
- [8] <http://www.northernjourney.com/opensource/newbies/newb020.html>. Introductie voor LaTeX-newbies die gebruik maken van linux.
- [9] <http://www.win.tue.nl/latex/documentation.html>. Uitgebreide Nederlandstalige documentatie over LaTeX.
- [10] <http://www.ntg.nl/maps/electromaps.html>. Elektronische versie van het ledenblad van de NTG: degelijke info.
- [11] <http://www.ntg.nl/whatislatex.html>. Wat is LaTeX?
- [12] <http://www.ntg.nl/mail.html>. Bekende site van de Nederlandstalige Tex Gebruikersgroep ;-).
- [13] <http://www.openoffice.org>. OpenOffice homepage.
- [14] <http://people.mech.kuleuven.ac.be/~bruyninc/>. Homepagina van Dhr. Bruyninckx, heftig voorstander van de open-source gedachte in Vlaanderen.
- [15] <http://people.mech.kuleuven.ac.be/~bruyninc/ictvisie.html>. ICT-visie van Dhr. Bruyninckx, pleidooi voor open formaten en open software.
- [16] <http://theory.uwinnipeg.ca/localfiles/infofiles/teTeX/>. Veel (Engels-talige) links en vooral info over TeX maar vooral over LaTeX.
- [17] <http://www.aaxnet.com/editor/edit029.html#office>. About conversion of Word documents.

community

BachTeX2003

as of old, and some more

Kees van der Laan
Hunzeweg 57
9893 PB Garnwerd Gr
The Netherlands
cg1@hetnet.nl
<http://home.hetnet.nl/~cgvanderlaan/>

abstract

A (partial) report of GUST's 11th meeting at Bachotek, Poland is given. It is incomplete because I could not understand most of the Polish contributions and I skipped the LaTeX day. It reflects just of one of the threads through BachTeX03's life. A question is raised: can the TeX-world follow with pdfTeX the evolving PDF standard?

keywords

BachTeX2003, GUST, Poland

Motivation

Nobody asked me to write a trip report, so I just did it for myself to increase awareness, to remember better what I have picked up, to ponder aloud about ideas which popped up, and not to let it fall into oblivion, not to let it go with the wind. Moreover, it was interesting to experience with this report as a pdfTeX e-paper, with links to WWW addresses, e.g. for the photos. This note is available at <http://home.hetnet.nl/~cgvanderlaan/notes/bachotex03.pdf> or with the .tex extension for the source, instead of the .pdf extension. A version with pages of screen size I will consider once I have created a BLUe Φ Output Routine for the purpose.¹

Around the meeting

This was the first time I did not really contribute to a BachTeX I attended, except for me being there.² Roughly 60 people participated, of which 6 from Holland, 3 from Germany, 1 from the UK, and 1 from France. No longer the Polish people consider BachTeX a family affair, also the Dutchies come with their spouses, children and pets. Michael is the example and Sveta + Beer joined me again. Taco and Bianca preferred to celebrate their 10th wedding anniversary at home.

On the (strenuous) way back,³ I had as driver plenty of time to think it over, while glimpsing the beautiful Polish May landscape, to let the highlights flash-back, to ponder about 'might-have-been's, about missed opportunities, to resmell the smoke of the bonfire,

1. Is it still needed with scrolling Acrobat? Yes, definitely, it is much nicer. Most likely I will use this approach in next year's report.

2. Well... maybe I contributed in the backwash by writing this report and making earlier BachTeX reports available on my site. It seems to me that I contributed similarly to TeX's life by my overview notes: What is TeX and MetaFont all about? of 1992, Catching up — pdf and html at the heart of 1998, and my last year's Professionals and amateurs. TeXnically I also contributed, IMHO, but it seems that these contributions have not been perceived as I hoped for, have not gained adherents as Ulrik mentioned to me. However, my paradigm series of notes is available at the TeX-Live 6 CD-ROM (see directory texmf/doc/paradigm), and who knows what more notes of mine are out there, because usually I'm not asked, whether I agree or consider it better to *un*publish. All my work published in MAPS is available on NTG's jubileum CD-ROM of 2001 which contains MAPS1–24, as a wealth of PDFs with search facilities. MAPS issues in HTML are available on the TeX Live CD-ROM.

3. Sveta in her trip report in Russian paid attention to this aspect, and some more.

to retaste the (smoked) sausages and (ashes-baked) potatoes, to allow the fire-spitters,⁴ redo their job on my retina, and... last but not least to hear the music, again.

Remarkable about the participants was that quite a few are retired T_EXies already, are no longer active with T_EX&Co. They only use T_EX&Co now and then for their pleasure, while in day-to-day life they are not allowed to use it or just don't use it for various reasons. This reminded me of one of my earlier thoughts about the lifetime of a T_EXie.

The logistics of the organization was in good hands: Jola and her team did as usual a good job, all went as smooth as smooth can be. Thank you! The lodging in the various bungalows spread along the lakeside was OK, the weather as usual, the food tasty⁵ — de GUSTibus non (T_EX) disputandum⁶ — and more than sufficient. However, there is one Polish soup — flaki (Eng: tripe), a gourmet of good repute — which I just can't swallow, but Beer really liked it.

The program/conference/organizing committees had it all well organized. The LaT_EX issues were concentrated on one day, so plain T_EXies like me had a day off. Thank you.

The proceedings were available at the spot as GUST bulletin 19. Thank you, Tomek, Stachek and Piotr! It is a pity that it is not complete, some authors could apparently not make it to submit in time.⁷

The lecture room compared to early BachoT_EXs has improved and is technically well-equipped with a 'beamer' connected to a PC to assist the lecturer.⁸ The computers in the computer room form an ethernet, and are connected by ASDL to the internet.⁹ So, email and browsing the WWW was available for the participants and their families.

Well-done! Thank you Jerzy, Marek...

I should have hanged around more in that room and learn from the experience of those present. For example Ronald Kwee contacted expressions of interest site where he traced the origins of the famous 20%-80% rule.

New, since 1998, is also a bar/café with terraces, to facilitate socializing.

Poland seems to prosper. The countryside is beautiful, especially in May with all those blooming trees, lilacs, and you name it. Money machines in the streets are common and convenient. There is less police than 10 years ago on the roads near for us foreigners difficult to recognize village/built-up areas to check your speed.¹⁰

The next BachoT_EX will be special: on the 1st of May, Poland will officially join the EEC! Moreover, Poland traditionally celebrates its proclamation on the 3rd of May 1791 of the first modern constitution of all Europe! And for me personally, it will be 10 years ago that I visited BachoT_EX for the first time, and 1100₂ years ago that I visited for the first time Hanna, Janusz Bien, Włodek, Jacko and GUST avant la lettre.

4. Peter is the children's hero: his rockets captivated Maarten among others and with the bonfire Maarten was Peter's best assistant.

5. I love the Polski grip the Polish mushrooms.

6. The slogan on this year's T-shirt.

7. Jacko's second paper is not in there because the editors considered the documentation which comes along with the macros sufficient. I tend not to submit any longer. I freed myself from restrictions/deadlines and I put the material on my WWW site when I'm done with it, like this report. Editors can include the links after verification in their column. There are disadvantages to publishing on the WWW, I know, but I consider the freedom as an extra for the time being, and the way of doing in general for the future — and my T_EX&Co future started in 1998 — if not the way out when forced by costs.

8. Or the lecturer can just team up (wirelessly?) the beamer to his private laptop.

9. So, I downloaded my last years work from my site in order to put the notes on display, but... no way: my PDF files could be watched on the screen but not printed?!? After all, I'm happy that it did not work because the pdfbookmark macros contained some serious flaws, ☹.

10. A village/built-up area is/was defined by one or more houses nearer than 50 yards to the road?!?

Presentations

For the conference program visit <http://www.gust.org.pl/BachTeX/2003>. Nice was as usual the handy map.

Presentation tools

Nowadays pdfscreen¹¹ seems to be the tool to be used. Although it is handy to have the buttons on the screen I prefer (hidden) links, which don't disturb the contents. It seems that people like to show their new acquired T_EXnologies, nothing wrong with that, and forget about the main purpose to convey the matter. It must be said that pdfscreen is more modest than just Acrobat reader with its full control bar.

For me a hypertexted pdf file supported by the keyboard buttons for next, previous, end and begin and some hidden links, is enough.

Hans Hagen seemed to have realized this earlier, because in one of his screen-oriented documentations (MetaFun) he realized tiny, very tiny buttons almost hidden in the border, nearly invisible because of the coloured and changing modest passe-partouts. Jacko also practices this approach as can be witnessed from his presentations. Nice!

Lectures

I realize that I won't do justice below to the majority of Polish speakers by not mentioning them. I just could not profit from most of the lectures because they were in Polish (no English abstracts let alone summaries, but... the titles were in English, fortunately.). And... sigh, after so many years I only can approximately say: in (w), yes (tak), no (nie), OK (dobry), bad (zły), good morning/day (dzień dobry), good evening (dobry wieczór), sorry (przepraszam), thank you (dziękuję), don't mention it (niema za co), please (proszę), goodbye (dowidzenia), lake (jeziro), lecture room (sala wykładowa), open (otwarte), closed (zamknięte), key (klucz), how (jak), silent (ticho), sir (pan), dining room (stolowka), breakfast (śniadania), lunch (obiad), dinner (kolacja), menu (karta), beet soup (barszcz), sauerkraut and meat (bigos), mushroom (grzyb), bon appetit (smasnego), I love you (ja kocham ciwie). I can order a beer/vodka (edno piwo/edna wodka) or a cup of tea/coffee (herbata/kawia) and similar trivia in Polish. No way of reading Polish let alone understanding a lecture, despite my recently acquired knowledge of Russian.¹² I'm sorry.¹³ But... there is hope: I learned some more words: plit(ok), streszczenie and klesh,¹⁴ one diminutive Katarzyna, and from my guide the pronunciation of the combination of letters: sz pronounced as sch, cz as tsch, examples steszczin, streszczenie, rz as z, example Andrzej, ch as h, example chchałbym.

There is some good news, however. Zofia Walcsak coined the idea of teaching us Polish before, along and after the conference informally, but more structured than along the meals, social gatherings or during the guitars in the night. And why not a weekly-follow-up of lessons by email? Some sort of network academy? Splendid, we like that.¹⁵ So GUST why don't you give it a try?¹⁶

11. I could not spot it, nor its documentation, on the CTAN 2002 CD-ROMs nor on the T_EX-live 6 CD-ROM. A keyword facility similar to the one on NTG's MAPS 1-24 CD-ROM of 2001 would be useful.

12. I was pointed out to some dangers: zapomnjat means in Russian to remember and in Polish the opposite: to forget!

13. The simultaneous translations for a small group somewhere in the room is not sufficient either.

14. At home Beer suffered again from several ticks; since last year I go to Bachotek with a tickpincer.

15. Since Januari Sveta and I enjoy satellite TV next to the cable and we can watch Russian TV and... Poland I and II! Who knows what that will bring. I know, I know, no way of picking up a single word at the moment, but maybe there are some educational broadcasts for children?

16. Maybe it will attract more foreign participants ☺. My wishes? To pronounce the words correctly. To understand the songs and to be able to join the quire in at least one of the traditional Polish bonfire songs. Sveta even talked about a BachTeX hymn. Katia has promised to send me in PDF one of the Polish songs typeset in Latin Modern, of course, along with an English translation which I'll put on my WWW site, next to the already available Russian songs. Other practicalities are for example to be able to understand the

Installing PostScript fonts for T_EX, Janusz Nowacki

It was obviously meant for a PC environment, not aimed at people like me, the Mac-ies. I got it that he summarized the files involved — .tfm .pfb .enc .map (.fd .sty) — and what to do where, meaning what these files should contain, and where they should be stored and under what name. Moreover, he used BoP's TOIL, ie the **Type One Installing Utility**. I like it when people write down the results after one has found one's way through the mass.

So, the installation and (mixed) use of (new) fonts in T_EX is still a nuisance, not yet solved in an easy way, I presume?

Anecdote. Janusz has the reputation of not understanding nor speaking English, but since The EuroT_EX at Kerkrade, since his lecture supported by slides/e-paper in English, he has proven that Polish T_EX-ies improve on themselves. So Janusz rehearse on it another time, if it is not too cumbersome.

Latin Modern, or an abundant extension of the Computer Modern family by accented characters using MetaType1, Bogusław Jackowski

It was all about diacritical characters as extensions to the CM fonts. As usual Jacko's contributions are very well-prepared. It seems that he is a professional in the sense of G.E.Forsythe: 'a professional starts where an amateur ends.' He provided English copies of his e-pages, the digital follow-up of the old transparencies, slides or foils. Moreover, the underlying paper in the proceedings was in English. Thank you Jacko! His walk along history lane of character embellishments, well necessary extras, was apparently influenced by the lecture style of Andrzej Tomaszewski. Nice! These new Latin Modern fonts, geared towards Polish diacritical characters, will be released in the exotic Hawaii. Thank you Jacko, your BoP company and the sponsors: DANTE, NTG....

Polishing T_EX has a history already. I remember the beginnings where the precise positioning of the embellishments were at stake. Now it is about uniting elements, about composed characters as *one* integrated symbol, in order to get rid of the problems in scaling among others, I presume?

MetaPost macros for dashing of closed areas, Bogusław Jackowski

He made use of the fact that MetaPost allows multiple occurrences of the operator `withcolor` in a single drawing statement. Thanks to that property the operator `withcolor` can cleverly be redefined to perform hatching rather than filling. Nice! I have to study the details and (re)do it to get the hang of it. I guess that the macros are available on the GUST fileserver for T_EX&Co materials.¹⁷

Indexing in Polish textbooks, Włodek Bzyl

His approach to indexing — linear sorting on the fly — is interesting. We'll soon be able to watch the real-life example of the Polish version of the T_EXbook. One remark, however. Why not just improve on Knuth by doing the sorting as sidestep from T_EX,¹⁸ ie as a subtask controlled by T_EX, and not necessarily done by T_EX as I did?

Typesetting tables using Metapost, Piotr Bolek

An interesting contribution because he essentially introduced the concept of layers — as we know already from Adobe's Photoshop — to T_EX&Co.¹⁹ Volker Schaa communicated

information on the street about car parking, and when and where we should obey what speed limits.

17. The GUST fileserver was also subject of a BoF.

18. The magical `\write18` with functionality to have access to system commands from within T_EX, was in the early nineties opposed by... Knuth!

19. Well, not true. Knuth already in his OTR approach used the overlay idea for printing multi-columns. I guess that Hans created a ConT_EXt OTR in order to include (coloured/changing) backgrounds of all sorts, not in the least for his e-papers. In PS we had the old overprint of DRAFT or CONFIDENTIAL all over the page in a shade of grey. I know, I know, one of those other missed opportunities: I should have talked

that there are problems to come when the tables interact with for example the text around. Moreover, Volker mentioned that layers are *the* new issue of Acrobat 6, which triggered some thoughts with respect to the evolving pdf standard and pdfTeX, or better the niche for TeX in there.

TeX tools in Windows XP or how to bite a cactus, Pawel Jackowski

Yes, this is the first time I attended a TeX-lecture of Jacko's son. As far as I understood, he mentioned the confusing differences in how to use TeX under various OS's on a PC. How to cope with it — bite the cactus — I could not grasp. Funny, and well-done in the family tradition.

Typescripts: the ConTeXt way of combining font families, Hans Hagen

Interesting and powerful how Hans solved the problem of mixing fonts, while paying attention to the quality aspects. Apparently, the virtual fonts idea is not enough. As of yet I have no hands-on with ConTeXt, but I'll give it a try once I have a new computer, and will pick from Hans' brains by looking under the hood.

How to make Jerzy love TeX again?, Hans Hagen

This lecture was all about showing the power of PDF forms when used together with TeX&Co. The text is not entered through an editor but through a pdf form, which he believes is a user-friendly way of using TeX. The user doesn't have to be aware that TeX&Co are under the hood. Hmmm, I see the claim, but.... What are the applications suited for this approach? What are the limitations? Let us wait and see for Jerzy's answer.²⁰

Hans uses for the purpose: pdfTeX, Acrobat based GUI, ConTeXt macros, PERL and Ruby scripts, XML dataflows, a watched folder system, collection of Ruby modules, classes and applications. Impressive!

Hyperref package, Martin Schröder

A bit fast to my taste he summarized the possibilities of this package written by Sebastian Rahtz in the late nineties. I missed examples. I pity that he did not mention that PS pictures should separately be converted to PDF (or PNG), as he told me later privately. So, older scripts to be enriched by hyperrefs should also be adapted with respect to markup and the accompanying (PostScript, ...) pictures should be transformed into PDF (or PNG) as well.²¹

It is just a pity that I could not understand Grzegorz lecture 'Towards better quality of pdf files,' which I guess is related to the above and to the work I started last year, although I took the quality for granted.²²

to Hans about this and about a lot more.

20. By the way Jerzy's complaint is that TeX is not user-friendly.

21. As far as I understand, ConTeXt does support creation of PS pictures via MetaPost and inclusion of these in PDF e-paper on the fly.

22. Yes, yes, I agree I should have talked to him, this is one of the might-have-beens I realized too late. Equally well — another missed opportunity — I should have talked to my friend Radek, about his usage of the Titanium Mac, simply because I'm about to buy a G4 Mac, either the Quicksilver or the Titanium notebook. I also would have liked to ask him about audio→MP3 conversion, and maybe he could have demonstrated it, and I could have judged the results. The results I get are insufficient, maybe because my hardware is too limited. Erik, yes he uses a Titanium nowadays, could not demonstrate it either because it seems that iTunes 3 with respect to MP3 conversion offers you the possibility to burn a CD in MP3, and not the possibility to simply convert an audio file into MP3. iTunes 4 seems to offer what I need. Radek's wireless contacting the ethernet access-point through the built-in airport (extreme?) did not work on the spot, alas. The functionality is nice and I would have liked to watch its performance.

BoFs

One was about the GUST file server for T_EX&Co materials — o Polskiej Bibliotece Internetowej — in Polish, ©. This was in parallel with Jerzy’s BoF, so we were not left unattended.

The future of European LUGs, *Jerzy Ludwichowski*

Maybe there was a change in the program, because his guidance was all about fund-raising from ‘Brussels’ and what sensible projects we could define for the purpose.²³

Hmmm, I wish him all the success he can get, but I doubt it that he will succeed.

The reason?

We had great and appealing projects like NTS, and we failed!²⁴

Moreover, I think that we are too narrow-minded, too much involved with T_EX’s purpose and possibilities. We should have an eye for multi-media — oh yes I know it is a buzzword — but 10 years ago I already day-dreamed in this direction, see my What is T_EX and Metafont all about?

We could start a multi-media project in teaching, eg ‘Polish for foreigners’, with T_EX&Co somewhere under the hood.²⁵ Hans’ forms could be used as interface for drilling, such that one can only proceed when results of the tests so far are sufficient. This kind of computer-assisted multi-media training I watched in practice when Sveta was learning Dutch. It is the modern way of learning foreign languages.

The point I like to make is that next to typesetting one badly needs compact sound (and video) for e-paper on CD-ROM/DVD and WWW, but this is beyond T_EX’s purpose. For acceptable transport time on the WWW we need still better compression techniques and/or faster transmission possibilities.²⁶ MP3 as compression for audio is not perceived as good enough. The multi-media examples I like are art CD-ROMs such as ‘Escher Interactief’ and ‘Het mysterie Margritte — een surrealistische ontmoeting met Margritte,’ the various CD-ROM encyclopedia, the e-dictionaries with pronunciation options, and the ‘Wining-and-Dinings’ to learn Spanish cooking for example.

Another aspect why we are in a weak position with respects to grants is that volunteers-biased User Groups, are not the structures for making real advances. Developments are made! Definitely! Somebody,²⁷ sometime, somewhere at BachoT_EX made the remark that real progress can apparently only be made at an University where PHD-students may invent new techniques supervised by a professor with vision, like Knuth, in want for the results. Hàn Thế Thành, with his pdfT_EX, guided by Jiří Zlatuška, seems to be an example.

Workshops

As usual before and after the meeting tutorials were offered. Very useful for the newcomers and for those who want to embark on a new corner of T_EX&Co.

Andrzej on bookcover design, Chris on digital presentations tools, Janusz Bien on GNU emacs for LaT_EX users, Jacek Kmiciek on LaT_EX—a second step, and Stachek on configuring and fine-tuning of T_EX installations under windows and linux.

23. I’m sorry that I did not make notes of the projects Jerzy listed.

24. A less ambitious project was LaT_EX₃ which also seems to have slipped of the road. eT_EX and PDFT_EX, a nearly one man’s effort, are successful, though.

25. Is the functionality–Polish for foreigners on CD-ROM/DVD—already available? If not that can be granted, I’m sure.

26. Maybe, internet via satellite?

27. Was it Ulrik?

Social events

Bonfire

The social event is the traditional bonfire with sausages to be grilled and potatoes to be baked in the ashes, while the ‘guitars in the night’ are playing with the people around as quire.

Jacko and his family are really great, not only because of their guitar playing: their a capella is something you should have listened to at least once in your life.

It must be said that not only Polish songs are sung, but also Ukrainean, Russian, and... English! In the early BachTeX’s we had Phil’s share and last year David Kastrup treated us on some of Leonard Cohen. Then, I was very much surprised by Marek Czubenko, who did not only join in singing the Polish, Ukrainean and Russian songs but als those of Cohen!

Orienteing competition

Another off-off social event was the orienting competition. These competitions are quite popular in Russia and apparently in Poland. Pecular! Funny! Tuned for the ocasion, was how you could score. Not only by finding the spots but also by emptying the bottle of beer: bringing the empty bottle(s) yields 1 extra point for each bottle and a cap yields 2 bonus points. Nice, well-done! Thank you Piotr and ...

Annual assembly

I really can’t tell what it was all about, but big laughter there was. I trust the GUST membership to decide the right things, so my default in voting is to abstain. Two motions were tabled, explained in English, and after voting accepted with one obstention, ☺. To understand the words of a motion is one thing, to realize the implications is another.

Pictures, alas no sound nor video

What is the value of a BachTeX report without pictures, without sound?²⁸ From the last EuroBachTeX a wealth of photo galleries is available at <http://www.gust.org.pl/BachTeX/2002>. I guess that when you read this report the photos of the corresponding year 2003 will also be available.

Pavel Jackowski mentioned <http://www.jaroslaw.pl/festiwal>.²⁹

I don’t know about a site with Polish music (sound, lyrics (translated), but those who want some Russian–lyrics (with transliterations or translations)–are invited to visit my site and find some links to Russian (bard) music.

I wonder when we may enjoy the bonfire songs from GUST’s site.

Evolving PDF standard

At the moment we have PDF 1.5 with Acrobat 6 around the corner. Jacko’s octopussy model of several years ago with PS at the heart, is replaced by PDF as the essential format

28. One or two pictures won’t do, although as for all we know a picture is worth a thousand words. Moreover, I have to wait until I have ended my film in my camera and have the film developed, printed and digitally written to CD. (My digital Quicktake 200 is out of order, alas.) Pondering over an e-paper approach I don’t know as of yet what is the good approach compatible with T_EX&Co. If we also like to have video in T_EX&Co then we have to use Adobe’s Acrobat. Acrobat ≥ version 5 allows sound and video import, and I read that pdfT_EX offers facilities for this. So, we have either the old approach of just (digital) picture inclusion for books, reports and ilks or the CD-ROM multi-media approach. Interesting! For this report I only included links to WWW’s with photos (gallery of pictures), no sound as of yet, alas. For the music you still have to join a BachTeX, nothing can replace BachTeX-live.

29. It is about last years festival in the fall: pesni nashi koreni (songs of our roots).

to go to and from.

If we use one or more layers of PDF 1.5 for T_EX&Co then we can always go *back* from this PDF layer to T_EX, realizing the functionality of $.tex \leftrightarrow .pdf$, after another tool has been used to modify some other layer.³⁰

The complexity of cooperating tools is decoupled, as long as we don't flatten the layers. PDF 1.5 and Acrobat 6 are potentially very useful.

Adobe Nederland in their recent Acrobat 6 introductory seminar advocated the use of layers for languages: a PDF file may contain layers with English, Dutch, Polish... translations. Automatic translation just has started but the results are still very poor.

Maybe, the workflow $.tex \xrightarrow{\text{T}_{\text{E}}\text{X}} .dvi \xrightarrow{\text{DVIPS}} .ps \xrightarrow{\text{Distiller}} .pdf$ is better suited for keeping pace with the evolving PDF standard than the shortcut $.tex \xrightarrow{\text{pdfT}_{\text{E}}\text{X}} .pdf$.

In my e-paper paradigm notes I will exercise both approaches—with or without PostScript—summarize the advantages and disadvantages and come up with conclusions.

I wonder whether we can keep pace with the evolving PDF standard in pdfT_EX. Apple has the same problem because from OS X onwards the screen is PDF oriented.

Conclusions

As one can see it is no surprise that the BachoT_EXs are well-attended. All the T_EXnical ingredients are there and well-cooked. Social life is very cosy and the location excellent. Moreover, if the meal goes with some music it can't be but a success. This time was no exception. For me the BachoT_EXs are the off-off-TUG meetings, they substantiate the T_EX-life I like.

Acknowledgements

Thank you GUST for having invited me again,³¹ you have definitely switched me in T_EX&Co active mode again, although there is much activity in the world outside, the world beyond T_EX&Co's purpose. Thank you conference committee and speakers for the nice material you offered. Thank you Jola for all the care you gave us, especially for organizing the lunch package which paved our way back. Thank you photographers for making your photos available, such that I could incorporate in this report links to them.³² Thank you all those present for the nice and cosy atmosphere.

See you next year,³³ if not earlier. My case rests, have fun and all the best.³⁴

Appendices

Experience in writing this e-paper

I ended up with a hybrid, a mixture, of a classical paper and an e-paper, take for example the page size which is A4 biased and that photos are not included but just linked to, which adheres to the hypertext idea.

30. The functionality, meaning, we can just refresh the T_EX&Co layer(s) independently of (what we did with) the other layers.

31. But, as promised on the EuroBachoT_EX I would have come, nonetheless.

32. Not all photos linked to are taken at BachoT_EX2003, but IMHO that does not really matter for conveying the impression.

33. For next year I intend not only to contribute a lecture or two, but also try the near to impossible to familiarize myself with the flavour of Polish, and grasp the opportunity to talk to people, getting rid of the missed-opportunities syndrome. Then there is my long-standing wish to join the 'guitars in the night' on my ocarino donated by Grizina, some years ago... keep fingers crossed.

34. Errors in this note? Do drop me an email: <mailto:cgl1@hetnet.nl>.

Troublesome is the handling of links to WWW addresses. My macros have been adapted such that the links are also written to a file, in order to maintain a survey of what is used. There is no warning whatsoever when an address becomes obsolete and the link dangling. That is what modern general hypertexts entail, I'm afraid.

Photos I did not include but just hyperlinked to them. This entailed a separation of the text and coloured photos. Watched interactively, it adds context information to the photos which is added value compared to just a photo gallery.

A PDF file put in a directory on my site won't be searched by search engines for META information. The search engines just look for the META information provided in a HTML page.

It seems that my pdf_{TeX} version, based on PDF 1.2 according to the file info, contains a flaw with respect to apostroph's, where the apostroph disappears.

Used WWW links

P15 <http://home.hetnet.nl/~cgvanderlaan/>
P15 <mailto:cgl@hetnet.nl>
P15 <http://home.hetnet.nl/~cgvanderlaan/notes/bachotex03.pdf>
P15 <http://home.hetnet.nl/~cgvanderlaan/notes/bachotex03.tex>
P15 <http://www.gust.org.pl/EuroBachotEX/photos/PBolek/imagepages/image15.htm>
P15 <http://home.hetnet.nl/~cgvanderlaan/notes/overview.pdf>
P15 <http://home.hetnet.nl/~cgvanderlaan/notes/texandco.pdf>
P15 <http://home.hetnet.nl/~cgvanderlaan/notes/profsandamateurs.pdf>
P15 <http://home.hetnet.nl/~slmorozova/notes/polen03ru.txt>
P16 <http://www.gust.org.pl/EuroBachotEX/photos/KBazargan/imagepages/image2.htm>
P16 <http://www.gust.org.pl/EuroBachotEX/photos/PBolek/imagepages/image9.htm>
P16 <http://www.gust.org.pl/EuroBachotEX/photos/TSheibak/imagepages/image10.htm>
P16 <http://home.hetnet.nl/~cgvanderlaan/plaatjesmap/bungalow.jpg>
P16 http://www.gust.org.pl/BachotEX/2003/plan_bacho.html
P16 http://www.xs4all.nl/~rkwee/20030430bachotex/web/20030502Hanna_Kimia_Albert_Maarten_Kim_Peter_Sam_water_rocket_launcher_at_lake.jpg
P16 <http://www.gust.org.pl/EuroBachotEX/photos/PBudzik/imagepages/image5.htm>
P16 <http://eoi.cordis.lu>
P16 http://www.xs4all.nl/~rkwee/eurobachotex/web/20020503x_Stanislaw_x.jpg
P16 <http://www.gust.org.pl/BachotEX/2003>
P17 <http://home.hetnet.nl/~cgvanderlaan/notes/bachotex94.pdf>
P18 <http://www.gust.org.pl/EuroBachotEX/photos/MWolinski/imagepages/image10.htm>
P19 http://www.xs4all.nl/~rkwee/20030430bachotex/web/20030503Radek_Anna.jpg
P20 <http://home.hetnet.nl/~cgvanderlaan/notes/overview.pdf>
P21 <http://www.gust.org.pl/BachotEX/2002>
P21 <http://www.jaroslaw.pl/festiwal>
P21 <http://home.hetnet.nl/~cgvanderlaan/russian.html>

LaTeX

Toolbox

Wybo Dekker
wybo@servalys.nl

abstract

Nieuwe avonturen in TeX-land.

keywords

index, rules, tooltips, uitvullen

Tegen fraudeurs

Sander van Geloven schreef:

Weet iemand hoe ik een `\hrulefill` kan maken die iets hoger ligt, eigenlijk precies in het midden van de hoogte van een gewone kleine letter. Nu heb ik met `\hrulefill` dit _____

maar ik wil eigenlijk (maar dan tot het einde van de regel) dit _____

Het is om te voorkomen dat iemand op de hardcopy nog tekst na het einde van de regel kan plaatsen. Nu kreeg ik van Piet van Oostrum de volgende twee tips:

```
\def\mrulefill{\leavevmode\leaders%  
\hbox{-}\hfill\kern0pt}
```

Maar die resulteert in een onderbroken streepjeslijn die opgebouwd is uit een heleboel ‘-’ karakters:-----
De tweede tip:

```
\def\mrulefill{%  
\leavevmode\leaders%  
\hrule height 3pt depth -2pt\hfill%  
\kern0pt}
```

was precies wat ik wilde, een ononderbroken lijn op halve x-hoogte: _____

Nu heb ik dit nog aangepast tot:

```
\def\mrulefill{%  
~\leavevmode\leaders%  
\hrule height .6ex depth -2pt\hfill%  
\kern0pt}
```

Dan is er nog een spatie tussen het einde van de regel met tekst en het begin van de lijn: _____

Verder heb ik ook de hoogte aangepast om het er beter uit te laten zien en die met de fontgrootte mee te laten schalen.

Tenslotte: ik zou ook wel mijn statuten document willen veranderen in een statuten LaTeX-klasse, alleen doe ik dit liever samen met iemand die hier al wat meer ervaring in heeft om gelijk een goed bruikbare statuten.cls te maken voor ieders gemak. Bij deze ook hier een oproep aan mensen die hier een steentje willen bijdragen.

Noot Wybo:

Door de *ex* als eenheid te gebruiken zorg je dat de rule-afmetingen meeschalen met het font, maar dat moet je dan ook voor de diepte doen. De `height .6ex` genereert een `\hrule` die op de baseline ligt en een hoogte heeft die 0.6 maal de x-hoogte is. De `depth -2pt` verwijderd aan de onderkant 2pt, zodat, als 0.6 maal de x-hoogte kleiner is dan 2pt, de `\hrule` zelfs helemaal verdwijnt!

De oplossing is dus:

```
\def\mrulefill{%  
~\leavevmode\leaders%  
\hrule height .6ex depth -.5ex\hfill%  
\kern0pt}
```

zodat je `\hrule 0.55-0.45 = 0.1ex` breed wordt en met zijn midden op $(0.55+0.45)/2 =$ een halve x-hoogte terecht komt: _____

Cursieve indextermen

Maarten Wisse schreef:

Als ik dit compileer:

```
\documentclass{article}  
\usepackage{makeidx}  
\makeindex  
\begin{document}  
Ik doe dit in de hoofdtekst:  
\index{ambitus@\textit{ambitus}}  
en ergens anders:  
\footnote{%  
  \index{ambitus@\textit{ambitus}}  
  Een voetnoot met een indexterm}  
\printindex  
\end{document}
```

dan wordt een `.ind` file gegenereerd met daarin:

```
\begin{theindex}
  \item \textit{ambitus}, 1
  \item \textit{ambitus}, 1
\end{theindex}
```

met als gevolg dat de indexterm *ambitus* tweemaal in de index komt. Weet iemand een manier om dit te vermijden? Er wordt in de LC van alles gezegd over protecten van commando's binnen voetnoten, maar wat ik protect (footnote, index of textit) het effect is hetzelfde: twee index items in plaats van één.

Johannes Braams antwoordde:

Een manier om het te vermijden weet ik zo 1-2-3 niet. Wel kan ik het verklaren, wellicht dat dat een clue levert. De twee items worden door makeindex als verschillend gezien vanwege de spaties. Die spaties worden veroorzaakt door het \protect-ed commando mechanisme van de huidige LaTeX; die die zorgt er namelijk voor dat het commando \index expandeert naar \index_ (waarbij de _ een spatie voorstelt). In de meeste gevallen zijn die spaties irrelevant en ziet LaTeX de volgende keer weer het commando \index, expandeert naar \index_ en dan hebben we twee spaties tussen \index en zijn argument ...

Piet van Oostrum schreef:

Dit is een oud probleem. Zie LaTeX bug 1435. (Gevolgd door een citaat van dat LaTeX-bug-rapport, dat voor mij niet te volgen was, dus dat laat ik maar even weg ...) Maar Piet kwam ook met een minder obscure oplossing: Gebruik footnote.sty en genereer de voetnoot met:

```
\begin{footnote}
  \index{ambitus@\textit{ambitus}}
  Een voetnoot met een indexterm
\end{footnote}
```

Of, als alternatief: redefinieer \index met:

```
\let\origindex\index
\renewcommand{\index}[1]{\origindex{#1}}
```

Maar dit vereist dat je geen speciale tekens zoals % \$ en \ in je index entries gebruikt zonder daar een \ voor te zetten (zoals dat in \index wel kan).

Vervolgens kwam Piet met nog een derde oplossing, afkomstig van Jeremy Gibbons uit *TeX and TUG news (Hey, it works!)*: maak en gebruik een style file met daarin:

```
% plainfootnote.sty: incorporate plain
% TeX's trickery into \LaTeX's footnote
% macros, to allow \verb"..." within
% footnotes (the argument to \footnote is
% not read before it is executed)
```

```
\long\def\@footnotetext{%
  \insert\footins\bgroup
  \footnotesize
  \interlinepenalty
  \interfootnotelinepenalty
  \splittopskip\footnotesep
  \splitmaxdepth \dp\strutbox
  \floatingpenalty \@MM
  \hsize\columnwidth \@parboxrestore
  \edef\@currentlabel{%
    \csname p@footnote\endcsname
    \@thefnmark}%
  \@makefnmark{\rule{\z@}{\footnotesep}}%
  \ignorespaces}%
  \futurelet\next\fo@t
}
\def\fo@t{\ifcat\bgroup\noexpand\next
  \let\next\fo@t
  \else \let\next\fo@t\fi \next}
\def\fo@t{\bgroup\aftergroup\@foot\let\next}
\def\fo@t#1{#1\@foot}
\def\@foot{\strut\egroup}
```

Al Piets oplossingen bleken te werken, de tweede zelfs beter dan hij zelf verwachtte.

Tenslotte kwam *Johannes Braams* met de suggestie: probeer eens makeindex -c (als je tenminste makeindex gebruikt). Volgens Frank Mittelbach zou dat moeten helpen zonder ander truken te moeten toepassen ... Maar dat bleek niet te werken: de -c-optie comprimeert weliswaar meer dan één spatie tot één spatie, maar ten opzichte van nul spaties blijft er dan toch nog een verschil over.

Tenslotte, als ik dan als spuit 11 nog een duit in het zakje mag doen: met een nieuw index-commando vang je twee vliegen in één klap:

```
\newcommand{\ITX}[1]{\index{#1@\textit{#1}}}
```

want ten eerste wordt \index net als in Piets tweede oplossing nu altijd vanuit een ander commando (\ITX) aangeroepen, zodat in de .ind-file geen verschillen ontstaan, en ten tweede kan in plaats van

```
\index{ambitus@\textit{ambitus}}
```

nu eenvoudig

```
\ITX{ambitus}
```

worden gebruikt.

xml

Docbook In ConT_EXt, a ConT_EXt XML mapping for DocBook documents

What is Docbook In ConT_EXt?

Docbook In ConT_EXt combines two technologies that are widely used by authors of technical literature: the Docbook DTD and the ConT_EXt macro package for T_EX.

It is a ConT_EXt module that allows one to produce a typeset version of a Docbook XML file, in dvi or pdf format.

It takes a Docbook XML file as input for T_EX. ConT_EXt's built-in XML parser parses the file and applies ConT_EXt commands when it reads opening and closing tags. Which ConT_EXt commands are applied, and therefore how the output is formatted, is determined by the Docbook In ConT_EXt module.

Docbook

Docbook¹ has been available since the early 1990s. Over the years it has evolved into an extensive DTD for technical literature. Long ago extensive, customizable stylesheets² became available, first in DSSSL, later also in XSLT. The Jade program and the JadeT_EX macro package made it possible to run the DSSSL style sheets and print the output with high-quality free tools. This enabled one to author, format and print Docbook documents without expensive software tools. With the advent of XML and XSLT more free tools have become available.

These combined features have made the Docbook DTD the DTD of choice for technical literature. The Linux Documentation Project is one well-known project that switched over from a private DTD to the Docbook DTD. Due to this strong position, the toolset for working with Docbook documents is growing rapidly, see e.g. <http://www.miwie.org/docbookinfo.html>³.

How did it start and where is it now?

During EuroT_EX 2001 in Kerkrade I had become interested in using ConT_EXt because of the beautiful presentation styles used by Hans Hagen and several other speakers. While I was following the ConT_EXt email list, I also became interested in ConT_EXt's XML capabilities. These seemed so wonderful to me, that I *had* to understand how this could be done using T_EX macro programming. I started asking questions. Sometimes Hans answers such questions with the suggestion that one take up some or other project. So he suggested that I start an XML mapping for Docbook.

I really had other plans, but I was so intrigued with ConT_EXt's XML capabilities that I could not resist and gave it a start. As an added benefit, I would become more familiar with the Docbook DTD. When I started I certainly was aware that this would not be a small task. Docbook is such a large DTD, allowing its authors to use the hundreds of

1. <http://www.oasis-open.org/docbook/>

2. <http://sourceforge.net/projects/docbook>

3. <http://www.miwie.org/docbookinfo.html>

elements in innumerable combinations. But only while the project evolved did it become evident to me how large it really is.

Michael Wiedmann, who is interested in all possible tools to render Docbook documents, heard about the project soon after I started it. He made several contributions. His support and interest helped me to continue through the difficult phase when a project is no longer new, but you do not yet have anything really usable and you know all too well how much work still has to be done.

Now, a year later, I have some sort of an answer as to how it is possible to program ConT_EXt's XML capabilities in T_EX macros: Theoretically T_EX macro programming is complete (is it called NP-complete?). Hans Hagen is one of the few programmers who can turn this theory into practice.

I also have a working XML mapping for DocBook documents in ConT_EXt, which I call Docbook In ConT_EXt (DIC). It contains good layout instructions for a number of oft used elements in their more common combinations.

Running Docbook In ConT_EXt

Before one can typeset an XML file `myfile.xml`, one should create a T_EX driver file `myfile.tex`, which should look something like this:

```
\input xtag-docbook
\starttext
\processXMLfilegrouped{\jobname.xml}
\stoptext
```

Then T_EX is invoked as: `texexec myfile.tex` to get a dvi file, or as `texexec -pdf myfile.tex` to get a pdf file.

In the driver file `xtag-docbook` is the file name of the module. The XML document is input with the `\processXMLfilegrouped` command. The filename `\jobname.xml` is always correct provided the driver file and the XML file have the same base name.

Alternatively, one can always use the same driver file, in which the name of the XML file is changed each time.

The ConT_EXt documentation indicates that one can also run the XML file as `texexec -xmlfilter=docbook testxml.xml`. This will not work because the name of the Docbook In ConT_EXt module does not conform to ConT_EXt's naming conventions. It works if the module is renamed as `xtag-doc.tex`.

Customizing Docbook In ConT_EXt

A Docbook XML document is a normal ConT_EXt document. The commands that make up a ConT_EXt document are also at work when a Docbook XML document is processed. They are just one layer away from what the user sees. Therefore the output can be customized as for any ConT_EXt document with ConT_EXt's setup commands. The setup commands should be given *after* the Docbook In ConT_EXt module has been read, so that they override the default setup commands in the module. If you do not give additional setup commands, ConT_EXt's defaults are applied. This is an example of a driver file with ConT_EXt setup commands:

```
\input xtag-docbook
\setupindenting[medium]
\setupheadertexts[section][pagenumber]
\setupheader[leftwidth=.7\hsize,style=slanted]
\setuppagenumbering[location=]
\setupitemize[each][packed][before=,after=,indentnext=no]
```

```
\starttext
\processXMLfilegrouped{\jobname.xml}
\stoptext
```

Docbook In ConT_EXt defines a few setup commands and other customizations of its own.

Section blocks

ConT_EXt always applies pagebreaks around section blocks, and it treats the Table of Contents and the Index as chapters. This behaviour can be changed with the `pagebreaks` option of the `\setupXMLDB` command:

- `\setupXMLDB[pagebreaks=all]`: Default ConT_EXt behaviour.
- `\setupXMLDB[pagebreaks=sectionblocks]`: ToC and Index do not start a new page, and they are treated as sections. All other section blocks retain their default ConT_EXt behaviour.
- `\setupXMLDB[pagebreaks=none]`: In addition to the `sectionblocks` option, `bodymatter`, `appendices` and `backmatter` do not start a new page.

Titles

Titles are formatted with a command of the form `XMLDB\XMLparent title`, where `\XMLDBparent` should be replaced with the name of the element to which the title belongs, e.g. `XMLDBarticletitle`. These commands can be redefined. They take one argument, the title. For example, the article title could be redefined as:

```
\def\xmlDBarticletitle#1%
  {\startalignment[left]\bfb #1\stopalignment \blank}
```

The section titles can be customized with ConT_EXt's usual `\setuphead` command.

blockquote, epigraph and attribution

The elements `epigraph` and `blockquote` have their own setup commands `\setupepigraph` and `\setupblockquote`, which have the following options:

- `narrower`. Both `epigraph` and `blockquote` are formatted using ConT_EXt's narrower environment. The value of this option is a list of `left`, `right` and `middle` that is passed on to the `\startnarrower` command. See the ConT_EXt documentation for `\startnarrower` for the effect of these settings.
- `quote`. The value is `on` or `off`. When `on`, quotation marks are applied as with ConT_EXt's quotation environment.
- `command`. The value is a command or set of commands, which are applied at the start of the narrower environment.

The element `attribution` is customized with the command `\setupattribution`, which has one option: `command`. The value is applied at the start of the attribution.

More customizations

Customization has only recently obtained the attention it deserves. More setup commands like those for `blockquote` and `epigraph` will follow. The distribution contains a document `Customization.xml` which will contain an up to date description of the customization options.

Other tools for the same task

Docbook In ConT_EXt is not the only tool for typesetting a Docbook document. The canonical tool for typesetting any XML file is XSL + FO. An XSL stylesheet is used to define the desired output in terms of Formatting Objects (FO). The FO description can be thought of as a formatter independent layout description. Then an FO processor is used to produce actual printed output, on paper or as an electronic document.

XSL stylesheets for Docbook have been available for several years, written by Norman Walsh. They implement a large part of the Docbook elements—not all elements, that seems impossible. And they are extensively parametrized, so that users can customize many aspects without modifying the XSL code.

The objective of XSL + FO is: one stylesheet, many processors. Several FO processors are available, among which two free tools: FOP and T_EX. FOP is a dedicated FO processor, that produces output in PDF. It is available from the Apache website⁴. T_EX can be used as an FO processor using David Carlisle's `xmltex` and Sebastian Rahtz's `passivetex` package, which runs under L^AT_EX.

ConT_EXt is a prospective FO processor. It already has an XML parser. Mappings should be defined for Formatting Objects, in the same way as I have done for Docbook In ConT_EXt.

Future plans

Currently, Docbook In ConT_EXt is not completely integrated with the ConT_EXt distribution. I have strictly used the ConT_EXt API wherever I could, and avoided to develop my own variants. But I have preferred to develop this module in separation from the development of ConT_EXt itself. The time has now come to work on a better integration. I hope this can be achieved over the next year.

If good, customizable XSL stylesheets for Docbook exist, and if ConT_EXt could be an FO processor for the resulting output, then why would it be a good idea to spend so much effort on writing a special Docbook stylesheet for ConT_EXt?

In the ConT_EXt community the idea of a special Docbook stylesheet for ConT_EXt has been greeted with enthusiasm. Apparently, here the theory of one stylesheet for many processors succumbs to the practice that users prefer to work with their tools of choice. For a popular set of tools like Docbook and ConT_EXt users afford the effort of another style sheet. Such a style sheet is more manageable for them and running the required tools is easier.

On the other hand, until now, *I* have spent most of the required effort. And *my* answer tends to be: Maybe it is *not* the best way to support Docbook and XML in ConT_EXt. Maybe it would be more useful to work on FO mappings in ConT_EXt.

Over the past year I have set up this stylesheet. I have investigated the main structure of Docbook and come up with a way to map that to a ConT_EXt document. I have implemented a framework for the mapping. I have enjoyed doing all that, and my insight and skills in T_EX macro programming have increased immensely. But the time has come that others take this over, add mappings for more elements, add customizations, add new ideas. I plan to move forward to more generic work to support formatting of XML documents using T_EX as the typesetting tool.

Availability

Currently, Docbook In ConT_EXt is available separately from the ConT_EXt distribution, from my web site⁵. Michael Wiedmann's web page⁶ with Docbook tools has a link to the Docbook In Context files.

Programming Docbook In ConT_EXt

ConT_EXt and XML

ConT_EXt can take XML documents as input. For that purpose it contains a non-validating XML parser, which recognizes XML tags as markup instructions. And it has an API

4. <http://xml.apache.org/fop>

5. <http://www.hobby.nl/~scaprea/context>

6. <http://www.miwie.org/db-context/index.html>

(Application Programmer's Interface) which allows one to define actions for those tags. This is called mapping XML tags to ConTeXt. A typical mapping instruction is

```
\defineXMLenvironment[element]{start action}{stop action}.
```

During the start and stop actions one has access to the attribute values of the element. For example, this is how one reads the `align` attribute of an `entry` element (in a table) and issues the corresponding setup command for ConTeXt's `TABLE` environment:

```
\doifXMLvar{entry}{align}%
  {\expanded{\setupTABLE[align=\XMLvar{entry}{align}]}}
```

ConTeXt's programming interface for XML mapping is robust. Rarely if ever does one get tangled in expansion problems. But, as is seen in the above example, timing the expansion remains an issue: The command to retrieve the attribute value, `\XMLvar{entry}{align}`, must be expanded before the setup command can be read by TeX. That is what `\expanded` does.

It is easy, is it not?

In principle, writing a mapping for an XML document in ConTeXt is simple. You state which ConTeXt commands you want to use for the start and stop of each element, and ConTeXt takes care of the rest. Practice is more complicated, certainly if you want to write a useful, extensible and customizable mapping for a complicated DTD. In the following sections I discuss a number of noteworthy features of the Docbook In ConTeXt mapping.

Encoding and language

An XML document declares its encoding in the `xml:encoding` declaration at the start of the document. ConTeXt supports several encodings, among which the XML default encoding `utf-8`. Correctly reading an encoding is one thing. Making all characters available that can be addressed by an encoding is quite another thing. Unicode and its `utf-8` encoding have brought all characters in the Unicode range, currently more than 50,000, within scope in a single document. At the moment many of these are mapped to 'unknown character'. Work is ongoing to bring more characters within reach of ConTeXt in a single document.

A Docbook document may declare its language in the `xml:lang` attribute of the document element. The Docbook in ConTeXt module contains at the moment translated strings for four languages: English, German, Dutch and Italian. These are used for automatically generated strings, such as the titles of the table of contents, the abstract, and the index.

Features for each element

Context stack

Because an XML document has a tree structure, each element in the document has a list of ancestors. I call that the context, or the context stack, which contains the ancestors from the document root to the current element.

An element may push itself onto the context stack when it starts, and pop itself when it finishes. In principle all elements should do so. In practice a number of elements omit this because they or their children do not use the context stack in their formatting.

During formatting, the context stack can be inspected with the following commands:

- `\XMLDBcurrentelement`: The current element's name.
- `\XMLancestor#1`: The name of the ancestor at level #1. The current element is at level 0.
- `\XMLparent`: The name of the current element's parent.

- `\the\XMLdepth`: The depth of the context stack.
- `\doifXMLdepth#1`: Execute the following instruction if the context stack has a certain depth.
- `\XMLDBprintcontext`: Print the context stack in the log file (mainly for debugging purposes).

ConT_EXt also defines the context stack. I have redefined it because ConT_EXt's implementation did not satisfy my plans. Later I have simplified my usage of the context stack. ConT_EXt's implementation may now be perfectly satisfactory, but I have not checked this.

ConT_EXt defines `\currentXMLElement`, which also holds the name of the current element. But it is only guaranteed to be valid while ConT_EXt reads the XML tag. Indeed, the mapping of some start tags in Docbook in ConT_EXt emit an `\egroup` command, which invalidates the value of `\currentXMLElement`.

Ignorable white space

XML has the interesting feature of ignorable white space. It can be used to give the raw XML document a nice formatting and make it fairly readable. (It did not exist in SGML. As a consequence, SGML documents may be practically unreadable in an ASCII editor.) For applications that read the DTD, this feature is rather clear: white space in elements whose content may only consist of elements, is ignorable. For example, when the content model of a section only contains paragraphs, all white space that surrounds the paragraphs is ignorable. Applications like ConT_EXt that do not read the DTD, must resort to other means to find out whether white space is ignorable or not.

I have introduced a feature that is similar to the mechanism used in XSLT. One can declare that an element preserves white space with the command `\defineXMLDBpreservespace#1`, and that it ignores white space with the command `\defineXMLDBstripespace#1`. For these declarations to work, the elements should be on the context stack, and they and their children should use the command `\XMLDBdospaces` as the last command in their start and end tags. `\XMLDBdospaces` has the effect of ignoring spaces following the XML tag if the current element has been declared to ignore spaces.

In practice this is only used by elements that would suffer if white space is not ignored. Note that T_EX itself already ignores a lot of white space, viz. all white space that it reads in vertical mode. In the example of white space surrounding paragraphs in a section, T_EX would do the right thing by itself.

The correct functioning of `\XMLDBdospaces` is rather subtle. The following is a generic element mapping:

```
\defineXMLenvironment[xxx]
  {\XMLDBpushelement{\currentXMLElement} \XMLDBdospaces}
  {\XMLDBpopelement \XMLDBdospaces}
```

The command `\XMLDBdospaces` in the start tag is executed while `xxx` is the current element. So it ignores white space if `xxx` has been declared to contain ignorable white space. But the same command in the end tag is executed *after* `xxx` has popped itself from the context stack. So its parent is the current element, and the command ignores white space if that *parent* has been declared to ignore white space. That is indeed exactly what we want, because the spaces following the end tag `</xxx>` are in the parent's content.

There is a class of ignorable white space that T_EX refuses to ignore: blank lines are converted to `\par` commands by T_EX's input scanner, before we can tell T_EX whether white space is ignorable or not. Even this does not always matter to T_EX because T_EX discards empty paragraphs or paragraphs that consist of white space only. In the above example we could insert blank lines between the paragraphs without ill effect. But a blank line between the start tag of a footnote and its first paragraph has a notably bad

effect: it introduces a `\par` command between the footnote number and the start of the text, so that the footnote number is in a paragraph by itself.

Such harmful blank lines can only be removed by preprocessing of the XML document. I wrote a tool to do that. It is a SAX document handler written in Java, which removes all ignorable white space. I call it ‘Normalizer’, and it is available on my web site⁷.

The output of this tool is not only good for the ConTeXt mapping. Looking over it is informative for authors of XML documents. Every amount of white space that is left by the tool, is regarded as meaningful white space by XML parsers. Is that really what the author wants?

Every element

In principle every element should contain the following commands:

```
\defineXMLenvironment [xxx]
  {\XMLDBpushelement\currentXMLElement
   \XMLDBseparator \XMLDBdospaces}
  {\XMLDBpopelement \XMLDBdospaces}
```

That is, it pushes itself onto the context stack. It checks whether it should typeset a separator. And it checks whether it should ignore following white space. In its end tag, it pops itself from the context stack, and it checks whether its parent should ignore following white space.

The separator is used by such elements as `author`, which may generate a comma or the word ‘and’ between consecutive elements. By default it is set to `\relax`. A parent element should give it a suitable definition to be used by its children, and reset it to the default when it finishes.

Which element is next?

ConTeXt’s XML parsing is event based. This means that the parser generates events, such as the start or stop of an element, and calls the associated actions. During the actions one only sees the current event. One cannot look back at past events, except for the data that one saved. One can certainly not look forward to check which elements follow. In contrast, XSLT is tree based. That means that one can scan all elements, preceding and following, in the formatting commands of an element. Event-based parsing may present serious problems to the programmer.

Is there a title?

An abstract may but need not have a title. When there is no title, I want to print the default title ‘Abstract’. Because of the event-based nature of the parse, one cannot at the start of the abstract look forward to see if a title will follow. One can only try to find a future event at which one may safely conclude that there is no title if one has not yet seen a title.

In an abstract the optional title may be followed by three types of element, `para`, `simpara` and `formalpara`. When any of these elements is started, one may safely conclude that either the title has been seen or there is no title.

One solution is to save the title, and to redefine the mappings of each of these three elements, such that they output the title or the default title if there was no title. And then restore their default definitions for the following elements.

Another way to tackle this problem is to save the whole abstract and process it twice. In the first pass we check whether there is a title. During this pass, all output should be suppressed. In the second pass we first output the title or the default title if no title was

7. <http://www.hobby.nl/~scaprea/context>

found in the first pass, and then we output the content. Again this requires a redefinition of the three possible elements that may follow the title, so that they suppress their output at the first pass.

The third option is provided by T_EX itself, not by the XML mapping. We redirect the typesetting of the abstract into a vbox. At the same time we save the title in the variable `\XMLDBtitletext`, which removes it from the typeset content in the vbox. Then we output the saved title or the default title if there is no saved title, and next we output the vbox. This is the best option, and I use it.

In a section this solution would be more problematic: we run the risk of saving a large vbox. Working with options one or two is also not fun, because there are more elements to be redefined. I think the only viable alternative would be to work with `\everypar`, because `\everypar` is T_EX's low-level signal that there is new text. Fortunately, in a section a title is required, so I did not (yet) have to work out this problem.

This is an example of the problems that arise because in an event-based parse it is hard to determine if an optional element is not present. The following section presents an example of the problems that arise because in an event-based parse it is equally hard to determine when a certain group of elements is finished.

Sectioning

Like many systems, ConT_EXt partitions its document in frontmatter, bodymatter, appendices and backmatter (called section blocks). The section block governs such properties as the numbering of the chapters and sections. I use the end of the frontmatter to print the table of contents.

Docbook does not have the equivalent of section blocks. There is not a single element that contains the frontmatter, the bodymatter or the backmatter. Therefore I analysed the top-level structure of a docbook document, and divided the elements that may occur as top-level elements into frontmatter elements, bodymatter elements and backmatter elements. When the first top-level bodymatter element is seen, the frontmatter is complete and the bodymatter starts. Similarly for the backmatter.

For a book in Docbook the situation is rather clear: The bodymatter starts with the first `part`, `chapter`, `article` or `reference`.

For an article the situation is much more fuzzy. While I counted only 6 top-level frontmatter elements, I identified 60 top-level bodymatter elements. The situation is complicated by the fact that some of these 60 top-level bodymatter elements may occur in the frontmatter at a lower level. For example, `abstract`, `authorblurb` and `address` may occur within the element `articleinfo`, which itself is a main constituent of the frontmatter. They may also occur as top-level elements, in which case I consider them as part of the bodymatter.

The transitions between the other section blocks are fortunately more clearly marked. The complete analysis is contained in the documentation in the module itself.

The situation is programmed using the commands `\XMLDBmayensurebodymatter` and `\XMLDBmayensurebackmatter`. All top-level bodymatter and backmatter elements execute the appropriate command. These commands check if the element is a direct child of the document element, i.e. if the depth of the context stack equals 2, and if the corresponding section block has not yet been started. The current section block is kept in the variable `\XMLDBsectionblock`.

Specific elements

Tables

Docbook uses the CALS table model. ConT_EXt uses two different table models. One is the tabulate environment, which is based upon T_EX's `\halign`. It is quite sensitive to expansion timing errors. The other is the TABLE environment, also called natural tables. It

is a very powerful and flexible environment, with much customization possibilities using `\setupTABLE` commands. A special feature of this table model is that rows, columns and cells can be configured both before and after their content has been given, at any time before the end-of-table (`\eTABLE`) command.

Because ConT_EXt's natural tables have much similarities to CALS tables, the mapping is in principle very easy: a row corresponds to TR, an entry to TD, `colspec` elements can be mapped to `\setupTABLE` commands.

There are three main complications.

- The top, bottom, left and right frames of a CALS table are determined by the `frame` attribute of the table; the `rowsep` and `colsep` attributes of the corresponding rows and cells should be ignored.
- CALS tables can have multiple `tgroup` elements, each with their own number of columns, and their own alignment and frame settings (`colspec` elements).
- Each `tgroup` may have its own `thead` and `tfoot` elements, which may contain their own `colspec` elements.

These requirements have led to the following model: The table element generates a ConT_EXt table, i.e. the table float, using the `\placetable` command. Each `tgroup` element generates its own TABLE environment, i.e. the actual table.

The table is not opened by the start tag of the `table`, because at that moment the title is not yet known. Instead, it is opened by the start tag of the first `tgroup` (command `\XMLDBopentable`, which contains the ConT_EXt command `\placetable`). The start tag of each following `tgroup` typesets the previous `tgroup` (command `\XMLDBendTABLE`). Before typesetting, the left and right frames are set up. The start tag of the second `tgroup` also sets up the top frame. The end tag of the `table` does the same as the start tag of the next `tgroup` would do. In addition it sets up the bottom frame of the table, and closes the `vbox` of the `\placetable` command.

The rest is careful attribute processing, and issuing the required `\setupTABLE` commands at the right time. Attribute processing generates a lot of overhead, because both the attribute names and their possible values have to be translated from CALS to `\setupTABLE`. That makes the code somewhat less readable, but the logic is quite straightforward.

Issuing the required `\setupTABLE` commands is a precise work.

- The start tag of the first `tgroup` applies the `frame`, `colsep` and `rowsep` attributes of the table (`\XMLDBopentable`), so that they apply to all TABLEs in this CALS table. The start tag of each `tgroup` applies its own `align`, `colsep` and `rowsep` attributes, within its own TABLE environment.
- `colspec` elements of a `tgroup` apply their attributes to the whole column of this TABLE. The `colspec` elements in the `thead` and `tfoot` elements, on the other hand, must save their attributes (`\XMLDBsavecolspec`); they will be applied per entry in the `thead` and `tfoot`.
- row elements apply their attributes immediately to the whole row.
- entry elements first check whether they are in a `thead` or `tfoot`; if so, they apply the saved `colspec` attributes. Then they apply their own attributes. This order is important. ConT_EXt gives precedence to properties set up per cell over properties set up for the whole table or per row or column. But in this case we apply what was originally a column specification per cell, so we must take care of the precedence ourselves.

Revision history

The revision history contains a number of revisions. Each revision specifies one or more fields out of five possible fields. I wanted to represent this in a table which should only

contain those columns for which at least one revision specifies data. Programming this was my first challenge in this project.

Hans Hagen suggested the solution. The revision history is saved, and then processed twice.

For the first pass of the saved revision history, we define the revision fields such that they register themselves when they occur, but suppress all output. We also count the number of revisions, so that we will know which row must contain the bottom rule of the table. Now we know which fields occur and we can setup the table and output its header row.

For the second pass we define the revision element such that it outputs the row with the fields. So that the fields are output in the same order as in the header row, regardless of their order in the XML document, we first save the fields in a revision, and at the end tag of the revision we output the whole row in the desired order.

I worked this out both in ConT_EXt's tabulate environment and with its natural tables. I decided to keep the solution with the natural tables, because natural tables are more flexible and less prone to expansion errors.

This procedure demonstrates a powerful feature of ConT_EXt's XML processing: It is possible to save a node of the XML document with its subtree; in other words, the content of an element, complete with embedded elements, is saved in a variable without parsing. Later one can process the saved subtree as often as one likes. In between one is free to redefine the behaviour of the embedded elements. In T_EX's macro language this is quite normal behaviour :

```
\def\savevar#1{\def\var{#1}}
... % redefine \processvar
\processvar{\var}
... % redefine \processvar
\processvar{\var}
```

In other programming languages it is not nearly as easy. Saving a node with its subtree in a SAX content handler so that it can be processed later is not a trivial task.

It is a disadvantage of the above procedure that the code is not easily read, certainly not if one is not used to the procedure. Recently, I have discussed an alternative procedure using Giuseppe Bilotta's `xdesc` module. It would achieve the same result but make the programming more transparent. Another advantage would be that it is more easily customizable by the user.

Program listing and CDATA

I have spent an enormous amount of time on program listings. At first it seemed easy: ConT_EXt has a verbatim environment which suits our purpose.

Then it was pointed out to me that some program listings contain CDATA sections, which were not treated well by my solution. I realized that a program listing is not really a verbatim environment because it does not disable XML tags. I dived deep into ConT_EXt's verbatim environment and came up with a variant that supported two types of verbatim: one real verbatim for CDATA sections and one that did only line oriented layout for program listings. Moreover, it was nestable, so that it could deal with CDATA sections within program listings.

But it remained problematic to get it quite right. When the end of the CDATA section or of the program listing element was followed by text on the same line, this text was lost. And my white space tool did exactly that: put the following text right behind the end of the program listing element.

When I revisited the problem a few months later it dawned on me that the whole verbatim approach was wrong. Neither CDATA sections nor program listing environments have anything to do with T_EX's notion of verbatim. CDATA sections just disable XML

markup. They may occur anywhere in an XML document, and have no semantic meaning. Indeed, an XML parser does not even report whether CDATA sections are used in an XML document; it simply resolves them.

For the program listing I found a simple solution. It avoids scanning a whole line at a time, therefore it avoids scanning the text following the end of the program listing with the wrong catcodes in place. It uses `\obeylines` and `\obeyspaces` and it places struts at the start of a line to prevent the leading spaces to be discarded by T_EX's paragraph mechanism. That is all, and it does the job well.

Hyperlinks, URLs and external documents

Docbook documents mark hyperlinks with the `ulink` element; the url is contained in its `url` attribute. If we were writing HTML documents it would be easy: `<ulink url="URL">text</ulink>` would be translated to `text</href>` and the browser would do the rest.

But not such an easy solution in a PDF document. Links to PDF documents should be treated differently from links to other documents, and relative links to non-PDF documents are not allowed. Therefore, we have to analyse the URL and complete it if necessary.

In ConT_EXt strings can be split into parts with commands like

```
\beforestring string\at substring\to\var
```

which splits `string` at `substring` and stores the first part in `\var`. I use this and similar commands to check whether the URL has an authority (this is the term used by RFC2396, which specifies URIs; usually it is called the protocol, e.g. `http`) and whether it is an absolute or a relative URL. If a local file is specified, we also check whether it has the extension `pdf`. Links to local PDF documents are created using the ConT_EXt command `\useexternaldocument`, links to other documents use the ConT_EXt command `\useURL`.

A special problem is posed by URLs like `slashdot.org`. Is it a web server, or is it a file in the current directory? Cf. the URL `myfile.html`, which has exactly the same pattern. After the terminology of RFC2396 I call this abbreviated URLs. By default they are not recognized. Thus `myfile.html` is correctly linked to as a local document, while `slashdot.org` is incorrectly linked similarly. The user can switch recognition of abbreviated URLs on by setting `\XMLDBcheckabbrURItrue`, and can switch it off again by setting `\XMLDBcheckabbrURIfalse`.

Unfortunately, I do not know how to get the working directory in a ConT_EXt run, so that relative URLs are currently not properly completed.

Customization

For a long time I did not pay much attention to customization. Recently, I received requests to make a mapping for the `blockquote` and `epigraph` elements. Together with that request a discussion arose on the ConT_EXt mailing list about customization. As a consequence these two elements and their child element `attribution` have proper setup options, viz. the commands `\setupblockquote`, `\setupepigraph` and `\setupattribution`. I am sure that more such setup commands will follow.

The same discussion on the ConT_EXt mailing list touched upon attributes whose range of values is not constrained. An example is the `role` attribute of the `para` element. It is not possible to define actions for such attributes in the stylesheet, because the possible values are not known. The idea arose to put a hook in the stylesheet for the user's own formatting command. Something like `\attributeaction[para][role]`. The user could define such an action with something like `\defineXMLattributeaction[para][role]`.

The stylesheet should invoke the attribute action within a group in order to allow the user to change fonts etc. for this element. Therefore ConT_EXt cannot invoke the attribute action automatically, because it cannot know where it should do so. For example, some

mappings for the opening tag invoke `\egroup`; if the attribute action had been invoked automatically, its scope would be ended immediately. This idea has not yet been implemented.

I am not sure how far customization can go. Enabling extreme customizability would come down to defining a new language for describing the formatting of a Docbook document. This would go too far. On the other hand, customizability is a strong feature of ConT_EXt. It is not difficult to add customizability options to the stylesheet; ConT_EXt has some good commands for that.

Acknowledgement

The first versions of the mappings for several elements, a.o. the `ulink` element, were contributed by Michael Wiedmann. He also contributed the string literal files for English and German. Giuseppe Bilotta contributed the string literals file for Italian.

And of course, nothing of this would have been possible without Hans Hagen's ConT_EXt.

application

Drawing Message Sequence Charts with LaTeX

Sjouke Mauw
Computing Science Department
Eindhoven University of Technology
P.O. Box 513
NL-5600 MB, Eindhoven
The Netherlands
sjouke@win.tue.nl

Victor Bos
Software Construction Laboratory
Turku Centre for Computer Science
Lemminkäisenkatu 14 A
FIN-20520, Turku
Finland
v.bos@abo.fi

abstract

The MSC macro package facilitates LaTeX users to easily include Message Sequence Charts in their texts. This article describes the motivation for developing the MSC macro package, the features of the MSC macro package, and the design of the MSC macro package.

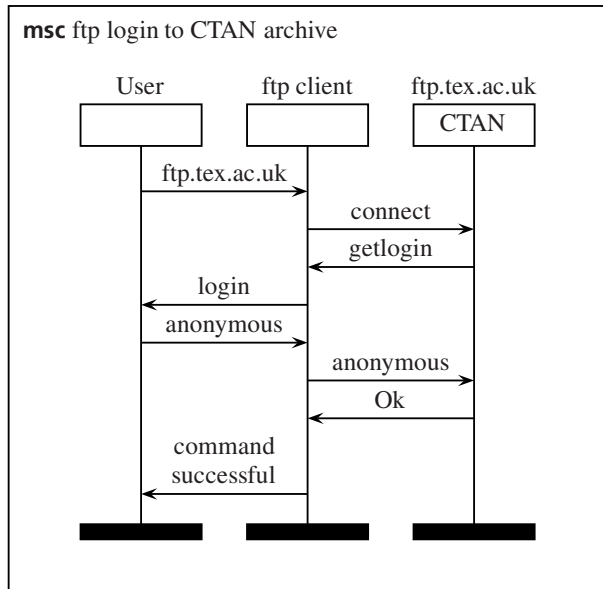


Figure 1. An ftp login to the CTAN at ftp.tex.ac.uk

Introduction

The Message Sequence Chart (MSC) language is a visual formalism to describe interaction between components of a system. The language is standardized by the ITU (International Telecommunication Union) in Recommendation Z.120 [IT01]. An introductory text on MSC can be found in [RGG96]. MSCs have a wide application domain, ranging from requirements specification to testing and documentation.

An example of a Message Sequence Chart is given in Figure 1. The MSC shows an ftp login session to a CTAN archive. Three players, called *instances*, are involved in the session: *User*, *ftp client*, and *CTAN* at location *ftp.tex.ac.uk*. The instances are denoted by vertical lines. Interaction between instances is denoted by labeled arrows. For instance, the arrow *ftp.tex.ac.uk* is a message from *User* to *ftp client*. Sending and receiving of a message are special types of *events*; each message has a *send* event and a *receive* event. Later we will see other types of events. Events occur on instance lines. Events are ordered in time and for each instance, time is supposed to run from top to bottom. Furthermore, the send event of a message never

occurs after the receive event of the message. For example, from Figure 1, we can derive that the *ftp.tex.ac.uk* message occurs before the *connect* message, because the receive event of the first message occurs before the send event of the second message.

In order to include MSCs in LaTeX documents, we have developed the MSC macro package. The current version of the MSC macro package supports almost the full MSC language as defined in the standard. In this article we will describe the motivation of the MSC macro package, the features of the MSC macro package, the design of the MSC macro package, and the limitations of the MSC macro package. This paper does not describe all features of the MSC macro package. For a thorough treatment of the MSC macro package we refer to the user manual [BM02b].

Motivation

Several commercial and non-commercial tools are available, which support drawing or generating Message Sequence Charts. However, these tools are in general not freely available and often not flexible enough to satisfy

all users' wishes with respect to the layout and graphical appearance of an MSC. Furthermore, they often do not allow the user to include LaTeX code in the MSCs. Another drawback of these tools is that quite often they restructure MSCs automatically. Though for simple MSCs this might be what the user wants, for more complex MSCs the result of automatic restructuring is usually not desired.

Therefore, people often use general drawing tools, such as *xfig* (see <http://www.xfig.org/>) to draw MSCs. However flexible this approach is, it has some drawbacks. First of all, general drawing tools have (and should have) a low level of abstraction; their interface is defined in terms of *coordinates, points, lines, polygons*, etc. To draw MSCs, the user would probably be more comfortable if the interface was defined in terms of *instances, messages, actions*, etc. For example, if you are drawing a message in an MSC using a general drawing tool, you would probably have to draw a *line* with an arrow head from a *position* (x_0, y_0) to a *position* (x_1, y_1) , instead of drawing a *message* from an *instance* i_0 to an *instance* i_1 of the MSC.

Another drawback of using general drawing tools is that they usually do not provide libraries of MSC symbols. Therefore, if you have to draw many MSCs, it will take much effort to get a set of consistent looking MSCs. Furthermore, if you want to change a parameter of the MSCs, e.g., the width of the instance head symbols, you would probably have to edit all MSCs manually.

For these reasons, we developed the MSC macro package for LaTeX. The macros in the package enable a textual representation of an MSC in a LaTeX source document. By compiling the LaTeX document into PostScript, a graphical representation of the MSC is generated.

The design requirements for the MSC macro package were:

1. The package should follow the ITU standard with respect to shape and placement of the symbols of an MSC.
2. The interface of the package should be at the right level of abstraction.
3. There should only be a limited amount of automatic restructuring and layout of the MSCs.
4. The appearance of (sets of) MSCs should be configurable by an appropriate set of parameters.
5. The MSC macro package should run on standard LaTeX distributions.

User interface

In this section we will briefly describe the user interface of the MSC macro package. We will do this by giving examples and showing the LaTeX code that produced the examples.

MSC environment MSCs are drawn in the `msc` environment. The syntax of this environment is

```
\begin{msc}[titlepos]{title}... \end{msc}
```

The title of the MSC is defined by the `title` parameter. The optional parameter `titlepos` determines the position of the title. By default it is `l` (left aligned). Other possible values are `c` (centered) and `r` (right aligned).

Instances Instances are declared with the

```
\declinst[*]{nn}{an}{in}
```

command. The starred version produces a *fat instance* which will not be discussed in this paper. The `nn` parameter defines a *nickname* of the instance. Nicknames identify instances and are used to draw messages and events. The `an` parameter defines the *above name* of the instance. This is the text to be placed above the instance head symbol (the rectangle at the top of an instance). The `in` parameter defines the *inside name* of the instance. This is the text to be placed inside the instance head symbol. Both the inside name and the above name may be empty.

Messages Messages are drawn with the

```
\mess[pos]{txt}{s}{r}[offset]
```

command. The optional `pos` parameter defines the position of so-called *self messages*: messages from an instance to itself. The default value of `pos` is `l` (to the left of the instance) and another possible value is `r` (to the right of the instance). The `txt` parameter defines the label of the arrow representing the message. The `s` parameter is the nickname of the instance on which the send event occurs, i.e., the nickname of the sender. The `r` parameter is the nickname of the instance on which the receive event occurs, i.e., the nickname of the receiver. The optional parameter `offset` defines the number of levels the receive event is shifted vertically with respect to the send event. Levels are discussed in the next paragraph. Offsets are useful if two instances send messages to each other and then wait for the messages to be received. For example, Figure 2 shows messages *a* and *b* between instances *i* and *j*. The receive event of message *a* occurs after the send event of message *b* and vice versa. Both messages have `offset = 2` in order to place the receive events two levels below the send events.

Levels The height of an `msc` environment is determined by the number of *levels* and a fixed amount of vertical space above and below the first and last level, respectively. Levels are created by the `\nextLevel[num]` command. The optional parameter denotes the number of levels to be added; its default value is `1`. Levels are used to order events in time. Recall that time runs from top to bottom, i.e., it runs from higher levels to lower levels. Events in the same level are drawn at equal vertical distance from the top of the MSC. The send event of a message will always be drawn in

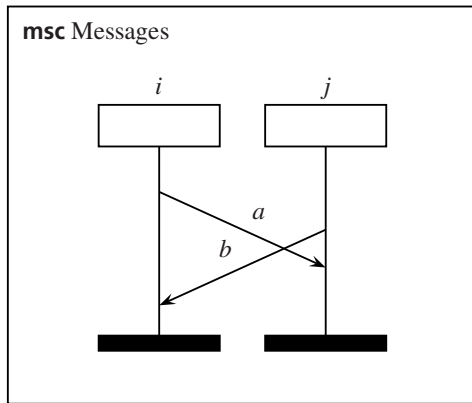


Figure 2. Using non-zero message offsets

```
\begin{msc}{Messages}
\declinst{i}{i}{}
\declinst{j}{j}{}
\mess{a}{i}{j}[2]
\nextlevel
\mess{b}{j}{i}[2]
\nextlevel[2]
\end{msc}
```

the current level. The receive event of a message can be drawn in another level using the `offset` parameter of the `\mess` command. Note that levels are not part of the MSC language, they are just an implementation means to draw MSCs.

Using the commands described so far, we can generate the msc of Figure 1. The LaTeX input to generate that MSC is given below. The length `\instdist`, used in the last `\mess` command, defines the distance between instances of an MSC and is one of the parameters to configure the MSC macro package. Here, it is used to create a `\parbox` that is 15% smaller than the distance between the instances of the MSC.

```
\begin{figure}[tb]
\begin{center}

\begin{msc}{ftp login to CTAN archive}
\declinst{usr}{User}{}
\declinst{ftp}{ftp client}{}
\declinst{ctan}{ftp.tex.ac.uk}{CTAN}

\mess{ftp.tex.ac.uk}{usr}{ftp}
\nextlevel
\mess{connect}{ftp}{ctan}
\nextlevel
\mess{getlogin}{ctan}{ftp}
\nextlevel
\mess{login}{ftp}{usr}
\nextlevel
\mess{anonymous}{usr}{ftp}
\nextlevel
\mess{anonymous}{ftp}{ctan}
\nextlevel
\mess{Ok}{ctan}{ftp}
\nextlevel[2]
\mess{\parbox[b]{.85\instdist}
```

```
{\centering command successful}}{ftp}{usr}

\end{msc}
\end{center}
\end{figure}
```

Actions Actions are events that can be used to model internal activity of a particular instance. Actions are defined with the `\action{txt}{nn}` command. The `txt` parameter defines the text to be placed inside the action symbol. The `nn` parameter is the nickname of the instance that executes the action. The action will be drawn at the current level with its top aligned with send events at the same level.

For example, suppose *CTAN* has to do some computations in order to determine if the anonymous login is allowed. The computation could be modeled by a *check* action, as depicted in Figure 3. The LaTeX code for the msc of Figure 3 is:

```
\begin{msc}{Action}
\declinst{ftp}{ftp client}{}
\declinst{ctan}{ftp.tex.ac.uk}{CTAN}
\nextlevel
\mess{anonymous}{ftp}{ctan}
\nextlevel
\action{Check}{ctan}
\nextlevel[2]
\mess{Ok}{ctan}{ftp}
\end{msc}
```

Regions Another way to model internal activity, or inactivity, is by using *regions*. Regions are defined by the `\regionstart{regtype}{nn}` and the `\regionend{nn}` commands. The `regtype` parameter defines the type of the region: activation, coregion (which will not be discussed in this paper), or suspension. The `nn` parameter is

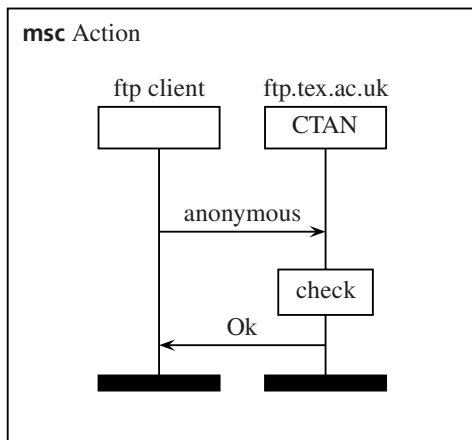


Figure 3. An MSC with an action

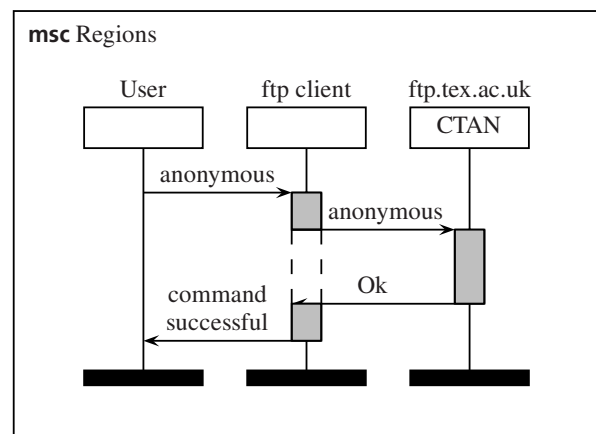


Figure 4. An MSC with activation and suspension regions

the nickname of the instance on which the region should be drawn. If an instance is active, e.g., doing some computations, this can be modeled by an *activation region*. If an instance is inactive, e.g., waiting for results, this can be modeled by a *suspension region*. For example, the computation of the *CTAN* could be modeled by an activation region. Furthermore, the *ftp client* is inactive during this computation, which could be modeled by a suspension region. Figure 4 shows the resulting MSC. The LaTeX code for the MSC of Figure 4 is:

```
\begin{msc}{Regions}
\declinst{usr}{User}{}
\declinst{ftp}{ftp client}{}
\declinst{ctan}{ftp.tex.ac.uk}{CTAN}
\regionstart{activation}{ftp}
\mess{anonymous}{usr}{ftp}
\nextlevel
\regionstart{suspension}{ftp}
\regionstart{activation}{ctan}
\mess{anonymous}{ftp}{ctan}
\nextlevel[2]
\mess{Ok}{ctan}{ftp}
\regionend{ctan}
\regionstart{activation}{ftp}
\nextlevel
\mess{\parbox[b]{.85\instdist}
{\centering command successful}}{ftp}{usr}
\regionend{ftp}
\end{msc}
```

Note that the space between the activation region of the *ftp client* and the *anonymous* message from the *ftp client* to *CTAN* is very small. In the next paragraph we will show

how redefining one of the *MSC parameters* can increase this space.

MSC parameters The *msc* macro package has almost 30 parameters to change the layout of MSCs. For example, the width of instances, the distance between instances, the distance between the head symbols and the *msc* frame, and the width and height of action symbols can all be changed. These parameters are represented by the LaTeX lengths `\instwidth`, `\instdist`, `\topheaddist`, `\actionwidth`, `\actionheight`, respectively. For instance, in the *msc* of Figure 4, the distance between the instances should be slightly bigger, in order to increase the space between the activation region of the *ftp client* and the *anonymous* message from the *ftp client* to *CTAN*. Figure 5 shows the same *msc*, but now the distance between instances is increased by 10%. The LaTeX code for this *msc* is the code for Figure 4 in which just after the line `\begin{msc}{regions}` the line `\setlength{\instdist}{1.1\instdist}` is included.

The location where an *msc* parameter is changed in the LaTeX source document determines its effect. Since the *msc* parameters are normal LaTeX macros or LaTeX lengths, the normal LaTeX scoping rules for these entities apply. For example, if a length parameter is changed outside any LaTeX environment, its effect is visible for all *msc* environments defined after the change. However, if it is changed inside an *msc* environment, its effect is only visible for that *msc*.

Since there are many parameters to configure the *msc* macro package, there are three predefined parameter settings to generate small, normal, or large *mscs*. The command `\setmscvalues{parset}` can be used to change the selected parameter settings. The `parset` parameter should be `small`, `normal`, or `large`. The default setting is `normal`.

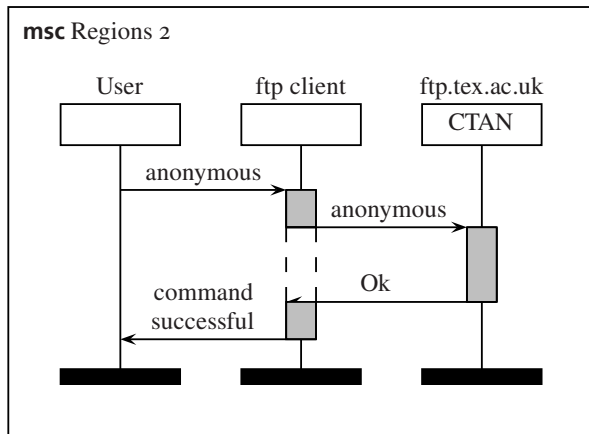


Figure 5. An MSC with larger distance between instances.

Implementation

In this section we will describe some aspects of the implementation of the MSC macro package.

Drawing MSCs In general, and as shown by the examples of the previous sections, an MSC consists of a number of vertically oriented instances that are connected by horizontally oriented messages. So, the width of an MSC is related to the number of instances and the height of an MSC is related to the number of (ordered) messages. Based on this observation, there are several implementations possible.

To define the width of an MSC, we could use an additional parameter of the msc environment. However, this strategy has some drawbacks. First of all, an extra parameter, the horizontal position, is required to declare instances. Furthermore, this parameter probably changes whenever a new instance is added to the left of an existing instance. Finally, the user should calculate the value of this parameter carefully in order to get evenly spaced instances.

Therefore, we chose to compute the width of an MSC based on the number of instances declared by the user and the user definable, `\instdist` length that defines the distance between instances. This decision does not violate requirement 3 of Section , no automatic structuring and layout, since the number of instances is under control of the user. Furthermore, the user can adjust the space to the left of the first instance and the space to the right of the last instance by redefining the length parameter `\envinstdist`.

The messages are partially ordered based on the relative position of their send and receive events on instances. We could have decided to provide commands to order the events and then let the package compute the final layout of the MSC. However, apart from the fact that this

computation is not trivial, this strategy fails with respect to requirement 3: no automatic structuring and layout.

Another strategy is to use an extra parameter of the msc environment to define the vertical size of an MSC. There are several drawbacks to this approach. First of all, the vertical size has to be computed. Secondly, commands to draw messages, actions, regions, etc., should have one or more additional parameter to indicate the vertical position at which they should be drawn. Finally, if a new message is to be added somewhere in the MSC, the vertical placement parameter of commands below the new message should probably be updated.

Therefore, we chose to only provide a command, `\nextlevel`, to advance the current height of the MSC. By increasing the current height between two messages, the partial order can be defined. Furthermore, one can easily add new messages to the MSC at any vertical position without having to change parameters of existing messages.

These decisions resulted in an msc environment in which the MSC is drawn in a top-left bottom-right fashion.

Nicknames As explained above, the MSC macro package uses nicknames to identify instances. If an instance is declared, the following attributes are associated to its nickname:

- The inside name,
- The above name,
- The width of the instance line,
- A flag indicating if it is a normal or a fat instance,
- The left, center, and right *x*-position of the instance,
- The *y*-position from which this instance still has to be drawn,
- The style of the instance line, and
- The style of the region of the instance.

The `\declinst` command defines the attributes using the following \TeX code pattern:

```
\expandafter\def
  \csname inst<attrnickname>\endcsname
  {<value>}
```

where `<attrnickname>` is the concatenation of the attribute, e.g., `abname` (above name), and the nickname and where `<value>` is the value of the attribute. For instance, the declaration

```
\declinst{usr}{User}{}
```

defines the following commands:

```
\instabnameusr, \instinameusr,
\instbarwidthusr, \instisfatusr, \instxposusr,
\instlxposusr, \instrxposusr, \instyposusr,
\instlinestyleusr, and \instregionstyleusr.
```

For each instance attribute, there is an internal command to read the value of the attribute. For example, to read

the value of the above name of instance `usr`, one should use `\msc@instabname{usr}`. For some attributes, like the current y -position, there is a command to change the value of the attributes. For example, to change the y -position of instance `usr` to the value y , one could use `\msc@setinstypos{usr}{y}`.

Underlying drawing engine The `msc` macro package uses the `pstricks` package, see [vZ93] or Chapter 4 of [GMS94], to draw lines, arrows, and frames. This package is now commonly available in LaTeX distributions, so relying on this package does not violate requirement 5. A drawback of `pstricks` is that it is incompatible with pdfLaTeX. Consequently, our `msc` macro package is incompatible with pdfLaTeX, too. However, there are other ways to generate pdf from LaTeX documents. One option is to first convert the `dvi` file into PostScript, e.g., using `dvips`, and then convert the PostScript file into pdf, e.g., using the `ps2pdf` utility included in ghostscript distributions (<http://www.cs.wisc.edu/~ghost/>).

Availability

The `msc` macro package is freely available at CTAN, see directory `macros/latex/contrib/supported/msc`, and at <http://www.win.tue.nl/~sjouke/mscpackage.html>. It is distributed under the *LaTeX Project Public License*, see <http://www.latex-project.org/lppl.txt>.

Documentation of the package consists of a user manual [BM02b] and a reference manual [BM02a]. These documents are included in the distribution.

Conclusions

The `msc` macro package enables users to include MSCs in LaTeX documents. Furthermore, the MSCs have a consistent layout that can be configured by an appropriate set of parameters. The package supports almost the complete ITU standard of the MSC language, including MSC documents and high level MSCs (which were not discussed in this paper).

1. The abstraction level of the `msc` macro package is as desired.
2. The user has full control over the relative position of instances, messages, etc.
3. Changing MSCs, e.g., adding extra instances or messages, is easy and does not require computations by the user.
4. The `msc` macro package is highly configurable. There are about 30 user definable length parameters and a small number of text parameters.

The developers of the `msc` macro package consider the package more or less complete. Therefore, the only changes to the package will be bug fixes and/or code documentation.

References

- [BM02a] Victor Bos and Sjouke Mauw. *A LaTeX macro package for Message Sequence Charts—Reference Manual—Describing MSC macro package version 1.13*, April 2002. Included in MSC macro package distribution.
- [BM02b] Victor Bos and Sjouke Mauw. *A LaTeX macro package for Message Sequence Charts—User Manual—Describing MSC macro package version 1.13*, April 2002. Included in MSC macro package distribution.
- [GMS94] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, 1994.
- [ITU01] ITU-TS. *ITU-TS Recommendation Z.120: Message Sequence Chart (MSC)*. ITU-TS, Geneva, 2001.
- [RGG96] E. Rudolph, P. Graubmann, and J. Grabowski. Tutorial on message sequence charts (`msc'96`). In *FORTE*, 1996.
- [vZ93] Timothy van Zandt. `Pstricks`, PostScript macros for Generic TeX. User's Guide, available at every CTAN site, (`CTAN:graphics/pstricks/`), 1993.

toepassing

Labels voor gevaarlijke stoffen met LaTeX

abstract

Volgens Europese regelgeving (67/548/EEC) is men verplicht om verpakkingen voor gevaarlijke stoffen te voorzien van labels die bepaalde informatie moeten bevatten. Met behulp van het `labels` package en een aantal in postscript geschreven pictogrammen is het mogelijk deze labels zelf te maken.

keywords

gevaarlijke stoffen, labels, 67/548/EEC, latex

Inleiding

Volgens de EG-richtlijnen moeten gevaarlijke stoffen en preparaten op de verpakking gekenmerkt worden. Als men zulke stoffen koopt, zijn ze door de leverancier van een label volgens de richtlijnen voorzien. Als men ze vervolgens overhevelt in een andere verpakking, moet men deze zelf van een dergelijk label voorzien.

Er zijn programma's verkrijgbaar voor het maken van zulke labels. Maar door het relatief kleine markt ervoor is deze software vaak niet zo goed van kwaliteit en relatief kostbaar. Het bedrijf waar ik werkzaam ben had voor het maken van deze labels een DOS programma, dat niet meer aan de praat te krijgen was onder windows. Een zoektocht naar alternatieven leverde alleen een veredeld tekenprogramma op waarbij je ook een speciale kleurenprinter moest kopen, of speciale voorbedrukte labels. Dit programma kon ook niet overweg met de database van stoffen die we voor het oude programma hadden aangelegd. Aangezien we al de beschikking hebben over een kleuren laserprinter, heb ik de labels in LaTeX 2_ε gemaakt.

Inhoud en opmaak van een label




Deze labels moet bepaalde informatie bevatten;

- handelsnaam van de stof of het preparaat
- naam van de gevaarlijke componenten
- gevarensymbolen
- R- en S-zinnen
- naam en adres van de leverancier

Deze informatie is te vinden op het veiligheidsinformatieblad volgens 93/112/EEG. Elke leverancier van gevaarlijke stoffen is verplicht een dergelijk informatieblad te beschikking te stellen.

De gevarensymbolen zijn ondermeer het bekende doodshoofd voor vergiftige stoffen en een kruis voor schadelijke en irriterende stoffen. Een overzicht van de symbolen en hun betekenis is weergegeven in tabel 1.

Tabel 1. Overzicht gevarensymbolen

Symbool	Kenletter	Betekenis	Symbool	Kenletter	Betekenis
	C	bijtend		Xn	schadelijk
	N	milieugevaarlijk		Xi	irriterend
	E	ontploffbaar		O	oxiderend
	F	licht ontvlambaar		T	vergiftig
	F+	zeer licht ontvlambaar		T+	zeer vergiftig

De term “R- en S-zinnen” staat voor de *risks* en *safety* aanbevelingen. Dit zijn een aantal standaard zinnen (vertaald in alle Europese talen) die aanduiden wat de risico’s van een bepaalde stof zijn en hoe je er veilig mee om kunt gaan. De R- en S-zinnen die van toepassing zijn op gedenatureerde alcohol zijn hieronder als voorbeeld weergegeven:

R11:	Licht ontvlambaar
R20/21/22:	Schadelijk bij inademing, opname door de mond en aanraking met de huid
R68/20/21/22:	Schadelijk: bij inademing, aanraking met de huid en opname door de mond zijn onherstelbare effecten niet uitgesloten
S7:	In goed gesloten verpakking bewaren
S16:	Verwijderd houden van ontstekingsbronnen – niet roken
S36/37:	Draag geschikte handschoenen en beschermende kleding
S45:	In geval van ongeval of indien men zich onwel voelt, onmiddellijk een arts raadplegen (indien mogelijk hem dit etiket tonen)

Een label voor een container tot 50 liter moet minstens 150 mm breed en 74 mm hoog zijn. Gewoonlijk staan maximaal twee gevarensymbolen links, en de rest van de tekst rechts. De symbolen zijn circa 3 cm in het vierkant.

Implementatie

De benodigde symbolen heb ik gevonden op het internet. Deze waren echter uitgevoerd als .GIF bestanden van lage resolutie en hetgeen er erg lelijk uitzag. Daarom heb ik de symbolen zelf geïmplementeerd in postscript, gevolgd door conversie naar PDF met `epstopdf`. De meeste symbolen zijn met de hand gedaan. De symbolen voor ontploffbaar en oxiderend zijn gemaakt met `autotrace`.¹

Oorspronkelijk is het label gezet als een geneste minipage omgeving. Als relatieve nieuwkomer in LaTeX kon ik deze opmaak niet krijgen zoals ik het hebben wilde. Daarom is het label nu uitgevoerd in een `picture` omgeving, met daarin een `\parbox` voor de tekst aan de rechterkant.

```
1 % -*- latex -*-
2 % $Id: ethanol.tex,v 1.3 2003/01/29 17:24:00 rsmith Exp $
```

1. <http://autotrace.sourceforge.net/>

```

3 %
4 % LaTeX sourcecode file for making 105x74 stickers for
5 % chemicals. Suited for containers up to 50 liters.
6 %
7 % Written by R.F. Smith <rsmith@xs4all.nl> in 2003 and placed
8 % in the public domain
9 %

```

Op de volgende regels worden de macro's gedefinieerd die de inhoud van de labels bepalen. Te beginnen met de handelsnaam, de gevaarlijke stoffen en eventuele codenummers

```

10 %%% Contents of the sticker; change these to suit you %%%
11 % \Chemname should contain the trade name for the substance
12 \newcommand{\Chemname}{Ethanol}
13 % \Contains describes the main ingredient
14 \newcommand{\Contains}{Bevat: ethanol, 5% methanol}
15 % \Codenumber is for the code your organization has given this
16 % material (if any). You can leave it empty.
17 \newcommand{\Codenumber}{1322 501 33801}

```

Vervolgens kunnen er maximaal twee symbolen met bijbehorende betekenis opgenomen worden. Het zou waarschijnlijk eleganter zijn om macro's te definiëren die een symbool en de bijbehorende tekst combineren. Maar in het opgenomen overzicht is duidelijk genoeg te zien welke combinaties gebruikt moeten worden.

```

18 % Possibilities for Upper-/LowerSymbol and Upper-/LowerText:
19 %      Symbol      English text      Dutch text
20 % E    explosive.pdf explosive      ontplofbaar
21 % O    oxide.pdf   oxidising        oxiderend
22 % F    flammab.pdf highly flammable licht ontvlambaar
23 % F+   flammab.pdf extremely flammable zeer licht ontvlambaar
24 % T    toxic.pdf   toxic            vergiftig
25 % T+   toxic.pdf   very toxic       zeer vergiftig
26 % C    corrosive.pdf corrosive        bijtend
27 % Xn   harmful.pdf harmful          schadelijk
28 % Xi   harmful.pdf irritant          irriterend
29 % N    environ.pdf dangerous for the milieugevaarlijk
30 %      environment
31 % none template.pdf
32 %
33 % The next four lines select the two symbols that can go on the
34 % sticker, and the accompanying texts. If you want to leave a
35 % symbol empty, use template.pdf
36
37 \newcommand{\UpperSymbol}{flammab}
38 \newcommand{\UpperText}{licht ontvlambaar}
39 \newcommand{\LowerSymbol}{harmful}
40 \newcommand{\LowerText}{schadelijk}
41

```

Vervolgens worden de R- en S-zinnen opgenomen. Deze zijn gezet als description elementen. Het zou mooi zijn om een macro te definiëren die automatisch de juiste zin genereert op basis van het nummer, bijvoorbeeld $\text{\R{20/21/22}}$. Maar dat is mij op dit moment T_EXnologisch te hoog gegrepen.

```

42 % Next are the R&S sentences. You should be able to find them
43 % on the MSDS for the substance. They are set as
44 % _description_ items.
45
46 \newcommand{\RSsentences}{
47 \item[R11:] Licht ontvlambaar
48 \item[R20/21/22:] Schadelijk bij inademing, opname door de mond
49     en aanraking met de huid
50 \item[R68/20/21/22:] Schadelijk: bij inademing, aanraking met
51     de huid en opname door de mond zijn onherstelbare effecten
52     niet uitgesloten
53 \item[S07:] In goed gesloten verpakking bewaren
54 \item[S16:] Verwijderd houden van ontstekingsbronnen --
55     niet roken
56 \item[S36/37:] Draag geschikte handschoenen en
57     beschermende kleding
58 \item[S45:] In geval van ongeval of indien men zich onwel
59     voelt, onmiddellijk een arts raadplegen (indien mogelijk
60     hem dit etiket tonen)
61 }
62

```

Tenslotte wordt nog naam en adres van de leverancier toegevoegd, en de datum waarop het bestand gecompileerd is.

```

63 % The name, address and phone number of the supplier should
64 % go here.
65 \newcommand{\Firma}{Chemproha -- Donker Duyvisweg 44, 3316 BM
66     Dordrecht\\ tel. 078-6544944}
67 \newcommand{\filedate}{\number\year-\number\month-\number\day}

```

Deze macro's bevatten alle variabele informatie van een label. Nu volgt de algemene code. Uitgegaan wordt van de standaard article class, met het labels package. Indien men labels in een andere taal wil maken, moet men natuurlijk een ander argument aan het babel package meegeven. Omdat ik meestal .PDF bestanden maak van dergelijke labels, heb ik het times package gebruikt om ervoor te zorgen dat er postscript fonts gebruikt worden.

```

68 %%% Normaly no changes required below here %%%
69 \documentclass[a4paper]{article}
70 \usepackage{times}
71 \usepackage[dutch]{babel}
72 \usepackage[latin1]{inputenc}
73 \usepackage{graphicx}
74 \usepackage{labels}
75
76 % Settings for the label package; 8 105x74 stickers on an
77 % A4 page.
78 \LabelCols=2
79 \LabelRows=4
80 \LeftBorder=0mm
81 \RightBorder=0mm
82 \TopBorder=1mm

```

```

83 \BottomBorder=0mm
84 \numberoflabels=8
85

```

De inhoud van het `\genericlabel` commando is de werkelijke definitie van de inhoud van de sticker. De maten die gebruikt zijn om de verschillende elementen van de `picture` omgeving te plaatsen zijn proefondervindelijk bepaald. De macro's die boven gedefinieerd zijn worden hier gebruikt. Dit deel van het bestand hoeft dan ook zelden aangepast te worden om een ander label te maken. Al komt het voor dat op regel 86 `\huge` vervangen moet worden door `\Large` indien de handelsnaam van de stof erg lang is.

```

86 \begin{document}
87 \genericlabel{%
88   \raisebox{-35mm}{
89     \setlength{\unitlength}{1mm}
90     \renewcommand{\sfdefault}{phv}\sffamily
91     \begin{picture}(104,74)
92       \put(5,40){\makebox(29,29){%
93         \includegraphics[width=29mm]{\UpperSymbol}}}
94       \put(5,36.5){\makebox(29,5){%
95         \textbf{\textsc{\scriptsize\UpperText}}}}
96       \put(5,8){\makebox(29,29){%
97         \includegraphics[width=29mm]{\LowerSymbol}}}
98       \put(5,4.5){\makebox(29,5){%
99         \textbf{\textsc{\scriptsize\LowerText}}}}
100      \put(35,62){\parbox[t][62mm][t]{60mm}{
101        {\huge\textbf{\Chemname}}\scriptsize
102        \textbf{\Contains}}\
103        \Codenumber\
104        \rule[2pt]{60mm}{.5pt}
105        \setlength{\leftmargini}{0pt}\vspace{-7pt}
106        \begin{description}
107          \setlength{\itemsep}{0pt}\setlength{\topsep}{0pt}
108          \setlength{\parskip}{1pt}
109          \RSSentences
110        \end{description}
111        \vfill
112        \rule[2pt]{60mm}{.5pt}\
113        \tiny\Firma\hfill\filedate
114      }
115    }
116  \end{picture}
117 }
118 }
119 \end{document}

```

Het resultaat ziet eruit zoals (verkleind) weergegeven in figuur 1.

Een LaTeX voorbeeld bestand en de bijbehorende postscript bestanden zijn te vinden onderaan op de software pagina van mijn website². Op het moment dat ik dit schrijf is de laatste versie `chemLabels-20030208.tar.gz`.

2. <http://www.xs4all.nl/~rsmith/software/>



Figuur 1. de opgemaakte pagina

metapost

Aligning METAPOST graphs in CONTEXT combinations

Introduction

For scientific plotting I like to use the Graph package by John Hobby within CONTEXT and when I have two or more separate graphs made I combine them into one figure with one figure caption. Combining is easy but aligning the graphs in a pleasing way required a trick.

Attempt 1

To demonstrate the problem I prepared two graphs plotting blood pressure and heart interval versus time (of course, it could be any signal) and added the result of spot measurements (BRS) to the graph. The vertical scales differ for each signal and these scales must be indicated in the graph. Each graph in isolation looks beautiful and a particularly nice feature is its seemingly endless scalability. Blowing the graph up from a design size of 5 cm width to 50 cm width or bigger is no problem.

CONTEXT then performs the combination of the graphs in one column aligned vertically using the `\startcombination[1*2]` command. The METAPOST pictures have been given an invisible outer box size, the so-called “bounding box” and this is a narrow fit to the total spaced used. METAPOST doesn’t care about centering the actual graph area in the bounding box. The graph areas, therefore, are not aligned vertically. See Fig. 1 left two panels.

Attempt 2

Adding the vertical scale values to the graphs, that differ in width, causes the miss-alignment. The plot areas are exactly 5 cm but the lettering sticks out on either side of it. This unbalancing effect could be annihilated by adding invisible texts of equal width, sticking out left and right of the graph area by an equal amount. An `\hbox` command comes to mind. An `\hbox` can be given a horizontal width. In CONTEXT a `\framed` command could also be used. Don’t try it, however, they don’t work unless they are filled with something that prints (thus not spaces) or as long as the empty `\framed` has a frame that’s not made invisible. Clearly, METAPOST refuses to adapt its bounding box to invisible matter. We, on the other hand, refuse to have extraneous matter printed.

Here is a solution that works, nevertheless. We let METAPOST draw a `fullsquare xyscaled(10cm,8cm)` centered at the midpoint of the graph area. This makes both graphs of equal width and height and alignment in a `\startcombination` is perfect. This is shown in Fig. 1 right two panels. The `fullsquare` is opaque. Anything plotted before it is drawn disappears behind it, anything plotted later is visible. It must therefore be the first plot command in the picture. In addition, it must be given a color and the color might be considered undesirable. The solution to that is to plot it with `color .9999white`. For all practical purposes this is as white as white paper is.

Here is the program for two curves within one graph:

```

\startreusableMPgraphic{outlier}
  picture Panel;          % .9999white for a white panel
  Panel:=image(fill fullsquare xyscaled(10cm,8cm) withcolor .9white);
  draw begingraph(5cm,5cm); % the plot area is 5 x 5 cm
  setcoords(linear,linear); % always set coordinates, or else
  setrange(210,100,230,125); % then set range
  gdraw (220,112.5) plot Panel; % center horizontal & vertical
  glabel.top(btex ibi (ms) etex,231,126);
  glabel.bot(btex time (s) etex,220,96);
  gdraw "sy13lb.ups" withpen pencircle scaled 1.5bp; % data plot
  autogrid(,); % always an autogrid, here empty
  setcoords(linear,linear); % hidden effect: resets scaling
  setrange(210,500,230,1500); % note a different vertical scale
  gdraw "in13lb.ups" withpen pencircle scaled .7bp; % data plot
  autogrid(otick.bot,grid.rt); % draw horizontal & vertical scales
  endgraph;
\stopreusableMPgraphic

```

And this is how two of them are used in a CONTEXT combination.

```

\placefigure[middle][fig:outlierb013]
  {Both left panels...
  }
  {\startcombination[1*2]
    {\reuseMPgraphic{outlier1}} {}
    {\reuseMPgraphic{outlier2}} {}
  \stopcombination}

```

Postscript

A true hacker would have thought of a solution in less time than it cost me. The manual that comes with Hobby's Graph package is written for experts, not for the common user. Graph's output is so accurate and scalable that one wants to use it anyway but experimentation is time consuming and when one must finish a production on time and within budget the level of experimentation required is bedeviling. The solution for that is to post a question with the CONTEXT user group.

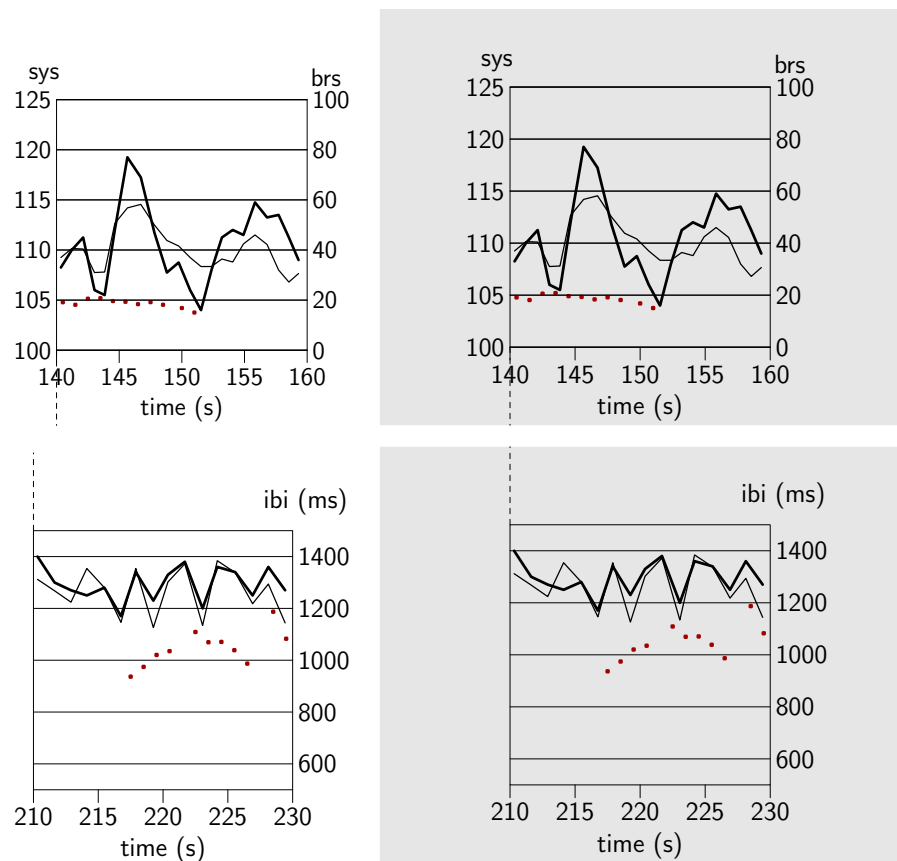


Figure 1 Both left panels on white background are clearly shifted with respect to each other as indicated by the dashed lines that do not meet. Right panels on a gray background show a correctly aligned system of diagrams. In all diagrams: the thick line represents systolic pressure variations in mmHg, over a 20 s observation window in a person (use 'sys' vertical scale), the thin line similarly is the time interval between heart beats in ms (use 'ibi' vertical scale), the crosses represent estimates of the sensitivity of blood pressure control by the bodies baroreflex expressed in ms/mmHg (use brs vertical scale) derived from the combined blood pressure and beat interval variability data which clearly move up and down in synchronism.

applications

Drawing a type-case in ConT_EXt

Willi Egger

abstract

There are different environments with which one can typeset tables; all of them have their advantages and disadvantages.

One of the recent problems I had to solve was to draw a typesetter's type-case from the lead-typesetting era. Since it looks like a table, I built the drawing in the `\bTABLE ... \eTABLE` environment.

Introduction

Formerly, when text was typeset by hand, fonts, e.g. letters, were stored in special drawers called type-cases (Dutch: letterkast, German: Setzkasten). These cases were strong wooden frames divided into four compartments by a wooden cross. Each compartment was divided into a series of boxes by thin wooden strips. The depth of such type-cases was small, normally not exceeding 3 cm. The layout of type-cases varied over time. Our drawing is an example of a three-quarter type-case popular in The Netherlands.

When typesetting text by hand, most of the letters are lower case. To facilitate picking the letters, lower case letters were located in the lower part of the type-case. From the size and location of the different boxes, one can deduce the number of single letters needed to assemble a text. The layout of a type-case was optimized for typesetting speed.

You might have realized that the expression of 'lower case' and 'upper case' (capitals) letters for the different fonts is linked to the location of the letters in the type-case. This is true for both English and Dutch: 'onderkast' en 'bovenkast' (kapitalen).

Analysis of the type-case

While looking for a straightforward approach to this drawing, it seemed best to build the type-case from four tables with the following characteristics arranged in a two by two matrix.

- All four tables have the same width.
- All table rows are 6em high.
- The width of the columns of the upper and lower left tables have a relation of 1.75: 2.
- The two top tables each have four rows.
- The two lower tables each have eight rows. Part of the cells are combined rows and / or columns.

Building the tables

General setups for the tables

The row height and the alignment of text within the cells are identical for all cells. These dimensions are defined outside the `\bTABLE ... \eTABLE` environment.

```
\setupTABLE[row]
    [even]
    [align={middle,lohi},
    height=6ex]
\setupTABLE[row]
    [odd]
    [align={middle,lohi},
    height=6ex]
```

The table for upper case letters

A	B	C	D	E	F	G
H	I	K	L	M	N	O
P	Q	R	S	T	V	W
X	Y	Z	J	U	„	— —

```

\bTABLE
\setupTABLE[c]
    [1,2,3,6,7,8]
    [width=3em]
\setupTABLE[c]
    [4,5]
    [width=1.5em]
\bTR
    \bTD A \eTD
    \bTD B \eTD
    \bTD C \eTD
    \bTD[nc=2] D \eTD
    \bTD E \eTD
    \bTD F \eTD
    \bTD G \eTD \eTR
\bTR
    \bTD H \eTD
    \bTD I \eTD
    \bTD K \eTD
    \bTD[nc=2] L \eTD
    \bTD M \eTD
    \bTD N \eTD
    
```

```

    \bTD O \eTD \eTR
\bTR
    \bTD P \eTD
    \bTD Q \eTD
    \bTD R \eTD
    \bTD[nc=2] S \eTD
    \bTD T \eTD
    \bTD V \eTD
    \bTD W \eTD \eTR
\bTR
    \bTD X \eTD
    \bTD Y \eTD
    \bTD Z \eTD
    \bTD J \eTD
    \bTD U \eTD
    \bTD \char132 \eTD
    \bTD -- --- \eTD
    \bTD \char146 \eTD \eTR
\eTABLE
    
```

The special character table

É	È	Ê	Ë	ff	fl	fi	ffl	ffi	^Æ _æ	Ç]
á	é	í	ó	ú	â	ê	î	ô	û	* †	§
à	è	ì	ò	ù	ä	ë	ï	ö	ü	!	?
1	2	3	4	5	6	7	8	9	0	(-

```

\bTABLE
\setupTABLE[c]
    [even]
    [width=1.75em]
\setupTABLE[c]
    [odd]
    [width=1.75em]
\bTR
    \bTD \'E \eTD
    \bTD \^E \eTD
    \bTD \^E \eTD
    \bTD \^E \eTD
    \bTD ff \eTD
    \bTD fl \eTD
    \bTD fi \eTD
    \bTD ffl \eTD
    \bTD ffi \eTD
    \bTD {\tfx \AE \OE \ \ \ae \oe}\eTD
    \bTD \c{C} \eTD
    \bTD $$$ \eTD \eTR
\bTR
    \bTD \'a \eTD
    \bTD \'e \eTD
    \bTD \'i \eTD
    \bTD \'o \eTD
    \bTD \'u \eTD
    \bTD \^a \eTD
    \bTD \^e \eTD
    \bTD \^i \eTD
    \bTD \^o \eTD
    \bTD \^u \eTD
    \bTD $$$ \ \ \$\dagger$ \eTD
    \bTD \char167 \eTD \eTR
\bTR
    \bTD \'a \eTD
    \bTD \'e \eTD
    \bTD \'i \eTD
    \bTD \'o \eTD
    \bTD \'u \eTD
    \bTD \'a \eTD
    \bTD \'e \eTD
    \bTD \'i \eTD
    \bTD \'o \eTD
    \bTD \'u \eTD
    \bTD ! \eTD
    \bTD ? \eTD \eTR
\bTR
    \bTD 1 \eTD
    \bTD 2 \eTD
    \bTD 3 \eTD
    \bTD 4 \eTD
    \bTD 5 \eTD
    \bTD 6 \eTD
    \bTD 7 \eTD
    \bTD 8 \eTD
    \bTD 9 \eTD
    \bTD 0 \eTD
    \bTD $($ \eTD
    \bTD - \eTD \eTR
\eTABLE

```

The table with part of the lower case letters

&	b	k	d	e
ç		c		
g	l	m	n	
z				i
y	v	u	t	pasjes
x				

```

\bTABLE
\setupTABLE[c]
    [1,2]
    [width=2.625em]
\setupTABLE[c]
    [3,4,5]
    [width=5.25em]
\bTR
    \bTD \& \eTD
    \bTD[nr=2] b \eTD
    \bTD k \eTD
    \bTD[nr=2] d \eTD
    \bTD[nr=3] e \eTD \eTR
\bTR
    \bTD \c{c} \eTD
    \bTD c \eTD \eTR
\bTR
    \bTD g \eTD
    \bTD[nr=2] l \eTD
    \bTD[nr=2] m \eTD
    \bTD[nr=2] n \eTD \eTR
\bTR
    \bTD z \eTD
    \bTD i \eTD \eTR
\bTR
    \bTD y \eTD
    \bTD[nr=2] v \eTD
    \bTD[nr=2] u \eTD
    \bTD[nr=2] t \eTD
    \bTD[nr=3] {\tfx pasjes} \eTD \eTR
\bTR
    \bTD x \eTD \eTR
\eTABLE

```

The table with the rest of the letters

Due to the fact that the last row of this table has the height of two ordinary rows, this has to be setup so:

```
\setupTABLE[r][last][height=12ex]
```

w	dunne spaties	f	g	h	j	hel
s					;	:
o	p	ij	sup.	dunne spaties	spaties	
			$\frac{1}{2}$ pt	inf.	vierkanten	
a	r		,	.	kwadraat wit	

```
\bTABLE
\setupTABLE[c]
  [1]
  [width=5.25em]
\setupTABLE[c]
  [2,3,4,5,6,7]
  [width=2.625em]
\setupTABLE[r]
  [last]
  [height=12ex]
\bTR
  \bTD w \eTD
  \bTD[nr=2] {\tfx dunne spaties} \eTD
  \bTD[nr=2] f \eTD
  \bTD[nr=2] g \eTD
  \bTD[nr=2] h \eTD
  \bTD j \eTD
  \bTD hel \eTD \eTR
\bTR
  \bTD s \eTD
  \bTD ; \eTD
```

```
\bTD : \eTD \eTR
\bTR
  \bTD[nr=2] o \eTD
  \bTD[nr=2] p \eTD
  \bTD[nr=2] ij \eTD
  \bTD {\tfx sup.} \eTD
  \bTD {\tfx dunne spaties} \eTD
  \bTD[nc=2] {\tfx spaties} \eTD \eTR
\bTR
  \bTD {\tfx $\frac{1}{2}$pt} \eTD
  \bTD {\tfx inf.} \eTD
  \bTD[nc=2] {\tfx vierkanten} \eTD \eTR
\bTR
  \bTD a \eTD
  \bTD[nc=2] r \eTD
  \bTD , \eTD
  \bTD . \eTD
  \bTD[nc=2] {\tfx kwadraat wit} \eTD \eTR
\eTABLE
```

Assembly of the type-case

In order to present the drawing as a type-case, the four tables are put into a frame. To get the placement right, several steps are necessary:

First, two times two tables are placed into a `\hbox`. The horizontal placement is achieved by `\removeunwantedspaces\quad`. Vertically the two `\hboxes` are separated by a `\vskip of 2ex`.

Finally, to enclose the drawing in a frame, the whole is placed into a `\framed` command.

```
\framed
[frame=on,
 rulethickness=1pt,
 offset=-.5em,
 align=normal]
{\hbox{\getbuffer[Caps]
 \removeunwantedspaces\quad
 \getbuffer[Spec-Char]
 \removeunwantedspaces}
\vskip2ex
\hbox{\getbuffer[Lower-Case1]
 \removeunwantedspaces\quad
 \getbuffer[Lower-Case2]
 \removeunwantedspaces}}
```

A	B	C	D	E	F	G	É	È	Ê	Ë	ff	fl	fi	ffl	ffi	Æ	Ç]	
H	I	K	L	M	N	O	á	é	í	ó	ú	â	ê	î	ô	û	*	§	
P	Q	R	S	T	V	W	à	è	ì	ò	ù	ä	ë	ï	ö	ü	!	?	
X	Y	Z	J	U	„	—	’	1	2	3	4	5	6	7	8	9	o	(-

&	b	k	d	e	w	dunne spaties	f	g	h	j	hel
ç		c			s					;	:
g	l	m	n	i	o	p	ij	sup.	dunne spaties	spaties	
z								$\frac{1}{2}$ pt	inf.	vierkanten	
y	v	u	t	pasjes	a	r	,	.	kwadraat wit		
x											

This is a large table, and will scale with the fontsize (height definitions in ex and width definitions in em). However, in order to typeset this drawing to fit in a given typesetting layout one can use the following mechanism:

A	B	C	D	E	F	G	É	È	Ê	Ë	ff	fl	fi	ffl	ffi	^Æ Ë _æ ë	Ç	l	
H	I	K	L	M	N	O	á	é	í	ó	ú	â	ê	î	ô	û	* †	§	
P	Q	R	S	T	V	W	à	è	ì	ò	ù	ä	ë	ï	ö	ü	!	?	
X	Y	Z	J	U	„	—	’	1	2	3	4	5	6	7	8	9	o	(-
&		k					w									j	hel		
ç	b	c	d			e	s	dunne spaties	f	g	h				;	:			
g							o	p	ij	sup.	dunne spaties	spaties							
z	l	m	n			i				$\frac{1}{2}$ pt	inf.	vierkanten							
y							a	r											
x	v	u	t			pasjes							kwadraat wit						

Figure 1 Layout¹ of an ordinary (dutch) type-case

```
\placefigure
[here]
[]
{Layout
\footnote{\quote{{\sc BOEK}
{\em over het maken van boeken
({\sc BOOK} on making books)}},
H. van Krimpen, 1966}
of an ordinary (dutch) type-case}
{\externalfigure[Type-Case]
[type=buffer,
width=\textwidth]}
```

1. 'BOEK *over het maken van boeken* (BOOK *on making books*)', H. van Krimpen, 1966

typografie

Optisch uitvullen in de Maps

Siep Kroonenberg

In deze maps wordt voor het eerst gebruik gemaakt van optisch uitvullen met behulp van *protruding characters*, dat wil zeggen dat wordt gepoogd de rechterkantlijn er rechter uit te laten zien door tekens met horizontale uitsteeksels, zoals onder andere afbreekstreepjes, iets in de kantlijn te laten uitsteken.

Dit is een optie van pdf \TeX die niet aanwezig is in een klassieke \TeX . Hàn Thê Thàn gaat uitgebreid hierop in in zijn proefschrift.¹ Hans Hagen heeft er al eens over geschreven in de Maps,² maar nu wordt het eindelijk ook in de Maps in praktijk gebracht, althans in de meeste La \TeX stukken.

Om dit te activeren moet je deze optie eerst aanzetten:

```
\pdfprotrudechars=1
```

Een waarde van 1 betekent: aan maar doe het pas nadat de alinea in regels is verdeeld zodat als je om wat voor reden dan ook toch een dvi-bestand wilt maken, er geen tekstverloop optreedt. Andere mogelijke waarden zijn 0 (uit) en 2 (wel tekstverloop).

Je geeft per font aan hoever welke tekens in de linker- of rechterkantlijn moeten uitsteken. La \TeX heeft een perfect handvat hiervoor: de derde parameter van het `\DeclareFontFamily` commando of de zesde parameter van het `\DeclareFontShape` commando. Hier kun je allerlei initialisatiecode kwijt. Meestal staat daar niets:

```
\DeclareFontFamily{T1}{ptm}{}
```

Voor de Maps Times-en-Frutiger fontfamilie staat daar nu:

```
\DeclareFontFamily{LY1}{ptfs}{%
  \rccode\font ' -=120
  \rccode\font ', =100
  \rccode\font '. =100
  \rccode\font ': =50
  \rccode\font '; =50
  \lccode\font 96=100 % quoteleft
  \rccode\font 39=100 % quoteright
  \lccode\font 147=250 % quotedblleft
  \rccode\font 148=250 % quotedblright
  \rccode\font 150=150 % endash
  \rccode\font 151=100 % emdash
}
```

Elke regel definieert een linker (`\lccode`) of rechter (`\rccode`) protrude-faktor voor één teken uit één lettertype. In de `\DeclareFontFamily` en `\DeclareFontShape` macro's staat `\font` voor: het font waar we het nu over hebben. Het teken wordt aangeduid met zijn plaats in de encoding vector. Bovenstaand voorbeeld is voor texnansi-encoding; voor de meer gangbare T1-encoding zou dat moeten zijn:

```
\DeclareFontFamily{T1}{...}{%
  \rccode\font ' -=120
  \rccode\font ', =100
  \rccode\font '. =100
  \rccode\font ': =50
  \rccode\font '; =50
  \lccode\font 96=100 % quoteleft
  \rccode\font 39=100 % quoteright
  \lccode\font 16=250 % quotedblleft
  \rccode\font 17=250 % quotedblright
  \rccode\font 21=150 % endash
  \rccode\font 22=100 % emdash
}
```

Als je een echte fijnproever bent of een erg buitenissig lettertype hebt dan zal het wel eens de moeite lonen om per lettertype protrude-factors te definiëren, maar zover ben ik niet gegaan. Het is ook wel de moeite om protrude-factors te definiëren voor letters zoals de 'T' en 'W' die erg uitsteken.

Op het moment van schrijven maken vier stukken geen gebruik van protrude-factors. Kijk maar eens of je die eruit kunt halen.

1. Hàn Thê Thàn, Micro-typographic extensions to the \TeX typesetting system, Masaryk University, Brno, 2000.

2. Hans Hagen, Hanging punctuation, a pdf \TeX microtypographic extension, Maps 25, najaar 2000.

fonts

Installing fonts in LaTeX: a user's experience

Ferdy Hanssen

abstract

This paper presents a user's experience with installing fonts for use in LaTeX. It will be shown that it is not hard to make a standard Type 1 font work, if you use modern font installation software for LaTeX. All the steps necessary to install the example fonts will be shown. The example fonts used are Adobe Garamond from Adobe and Mrs. Eaves from Emigre.

keywords

fonts, LaTeX, user

Introduction

Installing fonts in LaTeX has the name of being a very hard task to accomplish. But it is nothing more than following instructions. However, the problem is that, first, the proper instructions have to be found and, second, the instructions then have to be read and *understood*. We will show that it is not hard, by sharing with you our experience with installing two commercially available font families, that work out of the box on most computer systems.

We will only deal with fonts in the so-called Type 1, or Postscript, format. Truetype fonts are not within the scope of this paper. Also so-called expert fonts, containing lots of special characters, will not be discussed. We confine ourselves to fonts containing the Latin alphabet, with accents used in Western Europe. Furthermore we have installed all fonts within a TeX Live 7 installation on the Linux operating system. But the procedures should be similar on any TeX system which adheres to the TeX Directory Standard.

A very good book on the subject of installing fonts for use with TeX and LaTeX is Alan Hoenig's "TeX unbound" [5]. We have not read it completely, but what we have read indicates this book goes into quite some depth on explaining the issues with installing fonts for use with TeX. Unfortunately it is a rather expensive book, and when you have spent a lot of money on acquiring a set of nice, new fonts, you do not want to spend another substantial amount of money on a 580 page book. And you definitely do not want to spend a few days reading and (trying to) understand that book to be able to use your fonts with your favorite typesetting software.

Fortunately, there is another document available. An excellent guide is Philipp Lehman's guide [7]. Maybe its best feature is that it is available for free. You can simply download it from CTAN [3]. It certainly is a very good guide to help you installing and using your fonts without having to read hundreds of pages.

Both Alan Hoenig and Philipp Lehman recommend the use of the programme `fontinst` to help you install your fonts. Before we tried to install the font Mrs. Eaves, available from Emigre [4], we had never used this programme before. But with the use of Philipp Lehman's "Font Installation Guide" it turned out not to be too difficult to use, at least when the installation of a simple Latin alphabet is all that is required. Philipp Lehman suggests to read at least the general, introductory bits of the `fontinst` manual [6], the *Fontname* scheme by Karl Berry [2] and the standard LaTeX font selection guide [1], all available from CTAN [3], to acquire a reasonable understanding of the material.

Installing Adobe Garamond

The standard font family Adobe Garamond, also known as Adobe's Type 1 font package number 100, consists of six fonts. These are called:

AGaramond-Regular *AGaramond-Italic*
AGaramond-Semibold *AGaramond-SemiboldItalic*
AGaramond-Bold *AGaramond-BoldItalic*

When you buy this package from Adobe, these fonts are what you get. Adobe provides nice instructions on how to install and use them, just not for TeX. Luckily for you and me, TeX and LaTeX support has already been taken care of.

Table 1. Renaming of Adobe Garamond files

Font name	Original file name	New <i>Fontname</i> file name
AGaramond-Regular	gdrg____.pfb	padr8a.pfb
AGaramond-Italic	gdi____.pfb	padri8a.pfb
AGaramond-Semibold	gdsb____.pfb	pads8a.pfb
AGaramond-SemiboldItalic	gdsbi____.pfb	padsi8a.pfb
AGaramond-Bold	gdb____.pfb	padb8a.pfb
AGaramond-BoldItalic	gdbi____.pfb	padbi8a.pfb

When you want to use a newly acquired font, it is important to always check first if ready-made support is available, to save you a lot of work. For many Type 1 fonts this is the case, and it can be found on CTAN [3] in the directory `/fonts/psfonts`. Adobe Garamond support has been written by Sebastian Rahtz and is available on CTAN in the directory `/fonts/psfonts/adobe/agaramon`.

All that is needed now are only two things. First, the actual font files need to be renamed according to Karl Berry's *Fontname* scheme. Second, all files need to be copied to the correct spot in the standard TeX directory tree. All of this is nicely documented by Philipp Lehman [7].

The renaming bit is easy in this case, as the fonts are actually listed literally in Karl Berry's *Fontname* [2]. In the current version, which dates from May 2003, they are listed on page 37. But the best way to look for them is by performing a file contents search on all files with file extension `map` in the `fontname` subdirectory of your `TEXMF` tree. The files, as supplied by Adobe, should be renamed according to table 1. We only need the actual font data files, with file extension `pfm`, in this case. Note that the font name we will use in TeX or LaTeX later is *pad*.

Then the files should be copied to the proper place. But, what is that proper place. Actually that is pretty straightforward. First of all, all locally added TeX things should be installed in your `TEXMFLOCAL` tree. Where this actually is, depends on your installation. With TeX Live, you can select this location when you install it. In this local tree, you should create a directory `fonts` (if it not already exists), with subdirectories `tfm`, `type1`, and `vf`. Within each of these subdirectories a directory with the name of the foundry should be created. For Adobe we use the name `adobe`. And within each of these three directories we need a directory with the name `agaramon`, the same as the name Sebastian Rahtz gave to his package. In the same way we also need a `dvips/config` and `tex/latex/adobe/agaramon` subdirectory in `TEXMFLOCAL`. So we now have a directory tree which contains at least the directories shown in figure 1.

We now need to simply copy all files with extension `tfm` to the `fonts/tfm/adobe/agaramon` directory, all files with extension `pfm` to the `fonts/type1/adobe/agaramon` directory, and all files with extension `vf` to the `fonts/vf/adobe/agaramon` directory. All files with extension `fd` need to be copied to the `tex/latex/adobe/agaramon` directory. And finally the supplied file `pad.map` needs to be copied to the `dvips/config` subdirectory.

To make sure the TeX engine can find all these files, we now need to update the search database. For the TeX Live system the actual command is `mktexlsr` or `texhash`, but yours may be different. This can be found in the documentation of your TeX distribution. After this step we need to enable the map file. Again, on TeX Live just run the com-

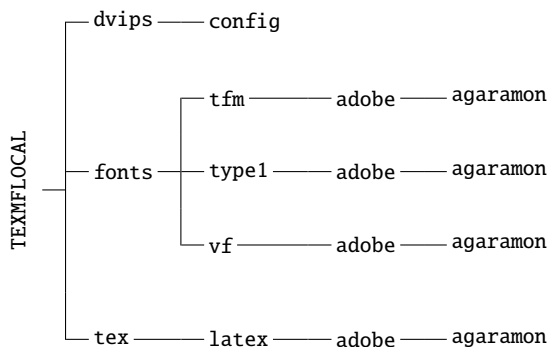


Figure 1. Directory tree for Adobe Garamond

mand `updmap -enable Map pad.map`. You should look in your documentation for the exact way of doing this.

Now all is set to use the newly installed Adobe Garamond font. You simply need to add the line

```
\renewcommand{\rmdefault}{pad}
```

to the preamble of your LaTeX document and you are in business. Or you can write a simple package by creating a file containing this line, followed by the line `\endinput`, naming it `agaramon.sty`, and placing it in the directory `tex/latex/adobe/agaramon`. Sebastian Rahtz did not include such a file in his package, although he did include it in some other packages.

Partially installing Mrs. Eaves

For the installation of Emigre's Mrs. Eaves we did not have the luxury of a ready-made package. Up to now we only have a very basic set of these fonts installed. The entire package consists of nine fonts, three of which contain only a large set of ligatures. The fonts in the base package are:

```
MrsEavesRoman      MrsEavesItalic
MrsEavesBold       MrsEavesSmallCaps
MrsEavesPetiteCaps MrsEavesFractions
```

The special fonts with ligatures are called:

```
MrsEavesJustLigRoman MrsEavesJustLigItalic
MrsEavesJustLigBold
```

The only fonts we have installed for use under TeX and LaTeX up to now are the first four. The *petite caps* font is a special variant of the *small caps* font, with even smaller capitals. The *fractions* font and the ligatures we have not yet looked at in detail.

We will provide you with the steps we needed for using

Table 2. Renaming of Mrs. Eaves files

Font name	Original file name	New <i>Fontname</i> file name
MrsEavesRoman	MEAVROMA.AFM	fevr8a.afm
MrsEavesRoman	MEAVROMA.PFB	fevr8a.pfb
MrsEavesItalic	MEAVITAL.AFM	fevri8a.afm
MrsEavesItalic	MEAVITAL.PFB	fevri8a.pfb
MrsEavesBold	MEAVBOLD.AFM	fevb8a.afm
MrsEavesBold	MEAVBOLD.PFB	fevb8a.pfb
MrsEavesSmallCaps	MEAVSMCA.AFM	fevrc8a.afm
MrsEavesSmallCaps	MEAVSMCA.PFB	fevrc8a.pfb

the first four within TeX, and we simply followed the first tutorial of Philipp Lehman's installation guide [7], which is suitable for installing simple fonts with a Latin alphabet. The first problem was that Mrs. Eaves is not mentioned in Karl Berry's *Fontname* scheme, so we had to come up with a new font name. We selected *fev*. The letter *f* stands for small, public foundries. The Emigre foundry seems to fit the bill here. The two-letter combination *ev* is unused up to now, and seems fitting for the name Mrs. Eaves. With the other guidelines for font naming we came up with the precise naming as seen in table 2. Note that the files with both extension *afm* and extension *pfb* need to be renamed. Renaming them in this way greatly simplifies the use of `fontinst` later.

We need to create a temporary directory with all renamed files in them. We also need a special driver file for `fontinst`. It has to look exactly like the one shown in figure 2 and it could be named `drv-fev.tex`. We then run `fontinst` in the temporary directory with the command `tex drv-fev`. This creates a lot of files.

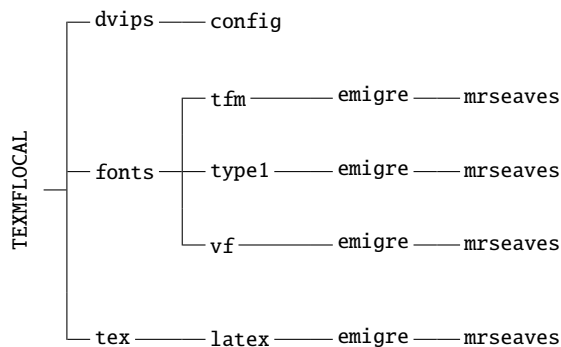
We now need to compile the metric and virtual fonts. This is done by running the programme `pltotf` on all files with extension *pl* and by running the programme `vptovf` on all files with extension *vp1*. This step creates even more files.

We now have almost all files needed to use Mrs. Eaves within TeX. Analogous to the Adobe Garamond fonts, we now need a directory structure to place our files. The necessary tree is shown in figure 3. Then we need to copy all files with extension *tfm* into the directory `fonts/tfm/emigre/`

```

\input fontinst.sty
\latinfamily{fev}{}
\bye

```

Figure 2. Driver file for Mrs. Eaves**Figure 3.** Directory tree for Mrs. Eaves

`mrseaves`, all files with extension *pfb* into the directory `fonts/type1/emigre/mrseaves`, and all files with extension *vf* into the directory `fonts/vf/emigre/mrseaves`. All files with extension *fd* should be copied to the directory `tex/latex/emigre/mrseaves`.

We also need a map file, so programmes like `xdvi` and `dvips` know what to look for. It can be tricky to write such a file, but using Philipp Lehman's tutorial it turns out to be not very hard to do. The map file we created can be seen in figure 4. We actually do not know if the value 0.167 as parameter to create a slanted font provides an aesthetically pleasing typeface. The designer obviously never intended such a font to exist. The map file should be named `fev.map` and placed in the `dvips/config` directory.

A simple package file, like the one shown in figure 5, makes using the font in LaTeX very easy. Note that we use the T1 font encoding. We simply follow Philipp Lehman's advice in this. Those encodings are not yet clear to us. This LaTeX package is called `mrseaves.sty` and placed in the directory `tex/latex/emigre/mrseaves`.

All that is needed to make the new font work now is running the command `mktexlsr` or `texhash`, followed by enabling the map file with the command `updmap -enable Map fev.map`. It must be said that it was a pleasant surprise to see it actually work after this.

```

\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{mrseaves}
[2003/09/03 v0.1 Emigre MrsEaves]
\RequirePackage[T1]{fontenc}
\RequirePackage{textcomp}
\renewcommand*{\rmdefault}{fev}
\endinput

```

Figure 5. Simple package for Mrs. Eaves

```
fevb8r MrsEavesBold "TeXBase1Encoding ReEncodeFont" <8r.enc <fevb8a.pfb
fevbo8r MrsEavesBold "0.167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc <fevb8a.pfb
fevr8r MrsEavesRoman "TeXBase1Encoding ReEncodeFont" <8r.enc <fevr8a.pfb
fevrc8r MrsEavesSmallCaps "TeXBase1Encoding ReEncodeFont" <8r.enc <fevrc8a.pfb
fevri8r MrsEavesItalic "TeXBase1Encoding ReEncodeFont" <8r.enc <fevri8a.pfb
fevro8r MrsEavesRoman "0.167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc <fevro8a.pfb
```

Figure 4. Map file for Mrs. Eaves

Conclusions

This paper describes our experience with installing two font families for use with T_EX Live. It is a lot easier than we thought at first. We owe, of course, many thanks to all the people who wrote the wonderful software, manuals and tutorials to make it so practical.

We only installed part of the Mrs. Eaves family. The intention is to make the others work within T_EX as well. As time permits we will look into this, partly as a learning experience, partly to be able to actually use that font within T_EX, and partly to share the resulting files and experience with you.

References

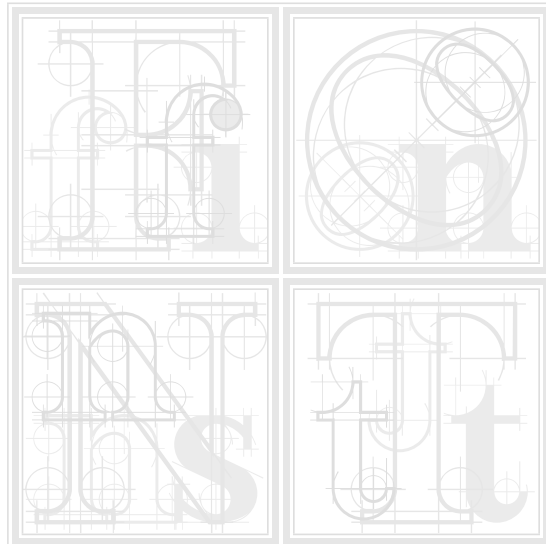
- [1] LaTeX3 Project Team. *LaTeX2e font selection*, Sept. 2000. Latest edition available on-line at <http://www.ctan.org/tex-archive/macros/latex/doc/fntguide.pdf>.
- [2] K. Berry. Fontname—filenames for T_EX fonts, May 2003. Latest edition available on-line at <http://www.ctan.org/tex-archive/info/fontname/fontname.pdf>.
- [3] Comprehensive T_EX Archive Network web site. <http://www.ctan.org/>.
- [4] Emigre web site. <http://www.emigre.com/>.
- [5] A. Hoenig. *T_EX unbound: LaTeX & T_EX Strategies for Fonts, Graphics, & More*. Oxford University Press, 1998. ISBN 0-19-509685-1.
- [6] A. Jeffrey and R. McDonnell. fontinst—font installation software for T_EX, June 1998. Latest edition available on-line at <http://www.ctan.org/tex-archive/fonts/utilities/fontinst/doc/manual/fontinst.ps>.
- [7] P. Lehman. The font installation guide, Aug. 2003. Latest edition available on-line at <http://www.ctan.org/tex-archive/info/Type1fonts/fontinstallationguide.pdf>.

Philipp Lehman schreef bijgaande handleiding voor het installeren van Postscript Type 1 fonts. Fontinstallatie zal nooit een fluitje van een cent worden, maar met deze handleiding bij de hand en de documenten die erin worden aanbevolen is het in elk geval niet langer een hachelijk avontuur voor helderzienden.

Wie de aangegeven stappen volgt, kan niet verdwalen. Lehmans handleiding is bovendien een lust voor het oog, fraai van functionele opzet en ook daarom drukken we het graag af in deze MAPS.

The Font Installation Guide

PHILIPP LEHMAN



AUGUST 2003

The Font Installation Guide

*Using Postscript fonts to their full
potential with Latex*

PHILIPP LEHMAN

VERSION 1.23

Copyright © 2002–2003, Philipp Lehman, lehman@gmx.net

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, version 1.2, with no invariant sections, no front-cover texts, and no back-cover texts.

A copy of the license is included in the appendix.

This document is distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose.

CONTENTS

INTRODUCTION	7
I THE BASICS	9
1.1 Renaming the files – 9 • 1.2 Using fontinst – 11 • 1.3 Installing the files – 14 • 1.4 Creating map files – 16 • 1.5 Using the fonts – 19 • 1.6 Computer Modern and T ₁ encoding – 21	
II STANDARD FONT SETS	25
II.1 The fontinst file – 25 • II.2 The <i>latinfamily</i> macro revisited – 29 • II.3 Map files revisited – 30	
III OPTICAL SMALL CAPS AND HANGING FIGURES	33
III.1 The fontinst file – 35 • III.2 The map file – 39 • III.3 The style file – 40 • III.4 Fonts supplied with Tex – 41	
IV THE EURO CURRENCY SYMBOL	43
IV.1 Uncoded euro symbol – 43 • IV.2 Euro symbol encoded as currency symbol – 46 • IV.3 Euro symbol taken from external symbol font – 47 • IV.4 Euro symbol taken from external text font – 52	
V EXPERT FONT SETS, REGULAR SETUP	55
V.1 Basic fontinst file – 55 • V.2 Verbose fontinst file – 56 • V.3 Inferior and superior figures – 59 • V.4 The map file – 63 • V.5 The style file – 64 • V.6 Using the fonts – 65	
VI EXPERT FONT SETS, EXTENDED SETUP	67
VI.1 The fontinst file – 67 • VI.2 Text ornaments – 74 • VI.3 The map file – 74 • VI.4 Extending the user interface – 75 • VI.5 A high-level interface for ornaments – 78 • VI.6 The style file – 79	
CODE TABLES	83
THE GNU FREE DOCUMENTATION LICENSE	87
REVISION HISTORY	95

INTRODUCTION

This guide to setting up Postscript Type 1 fonts for use with Tex and Latex is not systematic but task-oriented. It will discuss the most common scenarios you are likely to encounter when installing Postscript fonts. The individual tutorials collected here are not self-contained, though: the second tutorial will presuppose that you have read the first one and so on. All the tools employed in the installation process are documented well, the actual difficulty most users are facing when trying to install new fonts is understanding how to put all the pieces together. This applies to `fontinst`, the Tex font installation tool, in particular. Controlled by Tex commands, `fontinst` is a powerful and extremely flexible tool. While its manual documents all available commands individually, you will most likely wonder how to actually employ them after reading the manual. This is what this guide is about. Because of its concept, you will need the following additional manuals when working with it:

THE FONTINST MANUAL – Shipping as `fontinst.dvi`, the `fontinst` manual is the most important piece of documentation you will need when working with this guide since most files required for proper Postscript font support are generated by `fontinst`. You do not need to work through the sections explaining all low-level commands in detail, but make sure that you have read the more general parts and that you have a basic understanding of what `fontinst` is and what it does. If this manual is not included in your distribution, get it from the Comprehensive Tex Archive Network (CTAN).¹

THE FONTNAME SCHEME – Fonts used with Tex are usually renamed according to a dedicated naming standard, the Fontname scheme by Karl Berry. Take a look at the outline of the scheme as given in `fontname.dvi` and make sure you have copies of the individual map files at hand. These lists define names for a large number of commercial Postscript fonts. You will need them while working with this guide. If the documentation of the Fontname scheme is not part of your distribution, you can read it online² or download the complete package from a CTAN FTP server.³

THE LATEX FONT SELECTION GUIDE – It might be a good idea to read the Latex font selection guide as well before proceeding with the first tutorial. It provides an overview of the New Font Selection Scheme (NFSS). This scheme is not used during font installation, but it will help you to understand certain aspects of the installation process. This guide ships with most Tex distributions as `fntguide.dvi` and is also available in PDF format from CTAN.⁴ Feel

¹ <http://www.ctan.org/tex-archive/fonts/utilities/fontinst/doc/manual/>

² <http://www.ctan.org/tex-archive/info/fontname/>

³ <ftp://tug.ctan.org/tex-archive/info/fontname.tar.gz>

⁴ <http://www.ctan.org/tex-archive/macros/latex/doc/fntguide.pdf>

free to skip the chapter about math fonts as we are only going to deal with text fonts. Setting up math fonts is a science in its own right.

Please note that this guide was written with version 1.8 of fontinst in mind. On July 14, 2003, Lars Hellström has released fontinst 1.9 to the public.¹ The recipes proposed here should still work with the latest version, but they do not exploit the new features of the new release. I will try to update this guide as my time permits. The latest release of this guide can always be found at CTAN.²

Acknowledgments

I am indebted to Timothy Eyre for taking the time to proofread and comment on an earlier revision of the entire guide. I would also like to thank William Adams, Adrian Heathcote, and Adrian Burd for pointing out spelling mistakes.

¹ <http://www.ctan.org/tex-archive/fonts/utilities/fontinst/>

² <http://www.ctan.org/tex-archive/info/Type1fonts/fontinstallationguide.pdf>

TUTORIAL I

THE BASICS

This introductory tutorial serves two purposes. It covers the most basic installation scenario by explaining how to use fontinst's `\latinfamily` macro to integrate a small font family into a TeX system. By providing step-by-step installation instructions, it will also discuss the installation procedure as a whole. The later tutorials will focus on the more advanced capabilities of fontinst. Before we begin, let's take a look at an overview of the installation procedure:

STEP 1: RENAMING THE FONT FILES – First of all, we copy all Type 1 fonts (extension `pfb`) and the corresponding Ascii metric files (`afm`) to a temporary directory and rename them according to the Fontname scheme.

STEP 2: CREATING METRICS AND VIRTUAL FONTS – We will use fontinst, a font installer that works with Adobe font metric files in Ascii format (`afm`), to generate metric files and virtual fonts. Fontinst is normally not used interactively but controlled by a TeX file. Since the fontinst file is specific to a given font family, we need to write a suitable file for our fonts first and run it through TeX afterwards.

STEP 3: COMPILING METRICS AND VIRTUAL FONTS – Fontinst will generate font metrics and virtual fonts in a human-readable format which need to be converted to a machine-readable form afterwards. Hence we run all property list files (`pl`) created by fontinst through `pltotf` to create TeX font metrics (`tfm`) and all virtual property list files (`vp1`) through `vptovf` to create virtual fonts (`vf`) and the corresponding TeX font metrics for them.

STEP 4: INSTALLING THE FILES – We install all font metrics (`afm`), Type 1 font outlines (`pfb`), TeX font metrics (`tfm`), virtual fonts (`vf`), and font definition files (`fd`) into the local TeX tree. The remaining files are not required anymore and may be deleted.

STEP 5: CREATING MAP FILES – The fonts are now set up for TeX and LaTeX, but not for DVI and PDF drivers, which are configured separately. We create map files for `dvips`, `pdftex`, and, if a version of `xdvi` with native support for Postscript fonts is available, for `xdvi`. We install the map files and add them to the applications' configuration files.

STEP 6: UPDATING THE HASH TABLES – Finally, we run `texhash` to update the file hash tables used by the `kpathsea` search library.

1.1 Renaming the files

Users unfamiliar with fontinst tend to moan when introduced to the Fontname scheme for the first time. This file naming standard, which is also known by the name of its creator as the Karl Berry scheme, is often regarded as overly

complicated, cumbersome, unclear, and unmanageable. And indeed, it will appear somewhat cumbersome to anyone working with an operating system that does not impose silly limits on the lengths of file names. All of that is not the fault of its creator, however, but an inevitable result of the historical need to encode a complete font designation in a string of eight characters in order to cope with the limitations of the DOS filesystem as well as the ISO-9660 filesystem used for data CD-ROMs. The most important asset of the Fontname scheme is that it is the only formalized naming system widely used within the TeX community. Given the large number of files required to integrate a given typeface into a TeX system, installations without formal file naming would quickly get out of control. So, if the next couple of paragraphs should sound a bit cumbersome to you, you are in good company. Rest assured that after installing a few font families and watching your installation grow, you will understand the benefits of this scheme.

In order to understand the basic principles of the Fontname scheme, see the file `fontname.dvi` for an overview as well as excerpts from various map files. Browse the map files of individual vendors for the complete listings. When using the `\latinfamily` macro, strict adherence to the scheme is required. If you write a custom fontinst file using lower-level commands, the naming is technically up to you. It is still a good idea to stick to the naming system where possible. If a given typeface is not included in the map file for the respective foundry, take the foundry code from `supplier.map` and the code of the typeface from `typeface.map`. If the typeface is not listed at all, you will need to create a new code. This should be an unused one if possible. Try handling weight, variant, and encoding codes as strictly as possible. Foundry and typeface codes may be handled more liberally.

For large text font families, most font vendors do not put all fonts in a single package. They usually offer a base package containing upright and italic/oblique fonts plus an advanced package complementing the former. The advanced package will usually contain one of the following additional font sets: a set of optical small caps¹ and hanging figures,² a set of expert fonts,³ additional weights, or a combination of these sets. This package has to be purchased sep-

- ¹ ‘Optical’ or ‘real’ small caps, as opposed to ‘mechanical’ or ‘faked’ ones, are special glyphs found in a dedicated small caps font. They are preferable to mechanical small caps since they were actually drawn by the font designer. Mechanical small caps are generated by taking the tall caps of the font and scaling them down.
- ² While hanging or ‘old style’ figures have ascenders and descenders to blend in with lowercase and mixed case text, lining figures are aligned with the height of the capital letters (compare 1369 to 1369). Hanging figures are designed for use within mixed case text whereas lining figures are suitable for all uppercase text only. The latter also work well for applications like numbered lists and, since they are usually monospaced, for tabular settings.
- ³ ‘Expert’ fonts are complements to be used in conjunction with regular text fonts. They usually contain optical small caps, additional sets of figures, ligatures as well as some other symbols. Please refer to tutorial v for further information.

arately and can normally not be used independently in a sensible way. We will use Sabon as an example in this tutorial. The Sabon family offered by Adobe is split up into two packages. The base package contains upright and italic fonts (with lining figures) in regular and bold weights, while the so-called `sc & osf` package provides optical small caps and hanging figures. Hanging figures are also known as “old style figures”, hence the name `sc & osf`. In the first and the second tutorial we will deal with the base package only. Adding the `sc & osf` package to the base install will be discussed in the third tutorial. As we receive the package from Adobe or from a vendor, it contains the following files:

```
sar____.afm   sai____.afm   sab____.afm   sabi____.afm
sar____.inf   sai____.inf   sab____.inf   sabi____.inf
sar____.pfb   sai____.pfb   sab____.pfb   sabi____.pfb
sar____.pfm   sai____.pfm   sab____.pfm   sabi____.pfm
```

Of those files, we only need two types: the font metrics in Ascii format (`afm`) and the binary font outlines (`pfb`). We copy these to our working directory to rename them. In this case, finding the proper names is simple because the typeface is listed explicitly in `adobe.map`:

```
psbr8a      Sabon-Roman           A   088   sar____
psbri8a     Sabon-Italic         A   088   sai____
psbb8a      Sabon-Bold           A   088   sab____
psbbi8a     Sabon-BoldItalic     A   088   sabi____
```

The first column indicates the Fontname name and the last column the original name of the files as shipped by the vendor.¹ After renaming, we find the following files in the working directory:

```
psbr8a.afm   psbri8a.afm   psbb8a.afm   psbbi8a.afm
psbr8a.pfb   psbri8a.pfb   psbb8a.pfb   psbbi8a.pfb
```

We can now begin with the installation process.

1.2 Using fontinst

Since writing a fontinst file can be quite a time-consuming thing to do, fontinst provides a special macro which is able to deal with standard scenarios like this one. You can look up the `\latinfamily` command in the fontinst manual to understand what it does in detail. For our situation, it will suffice to say that it is able to recognize the standard fonts we provide by their file name – hence the need for strict adherence to the Fontname scheme in this case. Fontinst will create all metric and auxiliary files required by Latex without further directions in the form of lower-level commands. Therefore our fontinst file is as simple as it can get:

¹ The fourth column may also prove helpful: it indicates the number of the Adobe font package to which this font belongs. This number will save you a lot of time if you are trying to locate updated metric files for a font on Adobe’s FTP server since the files are sorted by package number there.

```

1 \input fontinst.sty
2 \latinfamily{psb}{}
3 \bye

```

After loading `fontinst` (1) we simply call the `\latinfamily` macro with the base of the file names (the foundry code plus the typeface code) as the first argument (2). The second argument is code to be executed whenever this typeface is used. This is often employed to suppress hyphenation of fixed-width typefaces by setting the hyphenation character to a non-existing encoding position. If we wanted to suppress hyphenation for this font family, we would call the macro like this:

```

2 \latinfamily{psb}{\hyphenchar\font=-1}

```

We save the file as `drv-psb.tex`, for example, and run it through `tex`:

```

tex drv-psb.tex

```

The `\latinfamily` macro will create metric files, virtual fonts, and auxiliary files for four different encodings: `TeX Base 1`, `OT1`, `T1`, and `TS1`. While `TeX Base 1` serves as the basis for virtual fonts using other encodings, it is usually not employed as such on the `Latex` level, although `\latinfamily` provides font definition files for the `TeX Base 1` encoded fonts as well.

The `OT1` encoding is a 7-bit legacy encoding solely suitable for text using the English alphabet only because it requires the use of composite glyphs when typesetting accented letters. These glyphs are inferior to the native glyphs provided by Postscript fonts. When using `OT1` encoding and typesetting the letter *a* with a grave accent, for example, `TeX` does not use the real glyph *à* as provided by the font because `OT1` discards all accented letters. This amounts to almost half of the glyphs found in common Postscript fonts. Instead, `TeX` will use the stand-alone grave accent and move it over the lowercase letter *a* to form a composite glyph. Apart from their inferior typographic quality, composite letters break `TeX`'s hyphenation algorithm so that words containing an accented letter are not hyphenated beyond this letter. Another problem with them is that they break searching for words containing accented letters in `PDF` files. In short, `OT1` should be considered obsolete unless you need the letters of the English alphabet only. But even in this case, `T1` encoding would be a sound choice.

`T1`, also known as Cork encoding, is a more recent text encoding suitable for a wide range of Latin scripts. Also known as Text Companion encoding, `TS1` complements `T1` by providing additional glyphs such as currency signs and other frequently used symbols like 'copyright' or 'registered'. `TS1` is never used as the main text encoding because it merely contains symbols. A user interface to the glyphs found in `TS1` is provided by the `textcomp` package.

When running the `fontinst` file through `tex`, `fontinst` will write a lot of messages to the terminal. These will include warnings about glyphs not being found, since a few glyphs defined in `OT1` and `T1` encoding are missing from the glyph set of our fonts:

```

(/usr/share/texmf/tex/fontinst/base/ot1.etx
Warning: missing glyph 'dotlessj'.
Warning: missing glyph 'lslashslash'.

(/usr/share/texmf/tex/fontinst/base/t1.etx
Warning: missing glyph 'perthousandzero'.
Warning: missing glyph 'dotlessj'.
Warning: missing glyph 'Eng'.
Warning: missing glyph 'eng'.

```

These warnings are normal, the missing glyphs are simply not provided by most Postscript fonts. In addition to that, you will most likely be lacking the ligatures ‘ff’, ‘ffi’, and ‘ffl’. This means that they will not be typeset as a single glyph but as a sequence of characters. There is no warning message in this case as fontinst will construct the ligatures using the single-letter glyphs at hand. You will usually find these ligatures in so-called expert fonts which complement the base fonts. Some foundries however, like FontFont, include them in the base fonts. Standard Postscript fonts should always provide the ligatures ‘fi’ and ‘fl’. The situation is worse for TS1 encoding since parts of it are rather exotic, defining glyphs not found in industry-standard fonts such as a ‘copleft’ symbol, or glyphs which should rather go in a dedicated symbol font such as arrow symbols:

```

(/usr/share/texmf/tex/fontinst/base/ts1.etx
Warning: missing glyph 'arrowleft'.
Warning: missing glyph 'arrowright'.
Warning: missing glyph 'tieaccentlowercase'.
Warning: missing glyph 'tieaccentcapital'.
Warning: missing glyph 'newtieaccentlowercase'.
Warning: missing glyph 'newtieaccentcapital'.
Warning: missing glyph 'blank'.
Warning: missing glyph 'hyphendbl'.
Warning: missing glyph 'zerooldstyle'.
Warning: missing glyph 'oneoldstyle'.
Warning: missing glyph 'twooldstyle'.
Warning: missing glyph 'threeoldstyle'.
Warning: missing glyph 'fouroldstyle'.
Warning: missing glyph 'fiveoldstyle'.
Warning: missing glyph 'sixoldstyle'.
Warning: missing glyph 'sevenoldstyle'.
Warning: missing glyph 'eightoldstyle'.
Warning: missing glyph 'nineoldstyle'.
Warning: missing glyph 'angbracketleft'.
Warning: missing glyph 'angbracketright'.
Warning: missing glyph 'Omegainv'.
Warning: missing glyph 'bigcircle'.
Warning: missing glyph 'Omega'.
Warning: missing glyph 'arrowup'.
Warning: missing glyph 'arrowdown'.
Warning: missing glyph 'born'.
Warning: missing glyph 'divorced'.
Warning: missing glyph 'died'.
Warning: missing glyph 'leaf'.
Warning: missing glyph 'married'.
Warning: missing glyph 'musicalnote'.

```

```

Warning: missing glyph 'hyphendblchar'.
Warning: missing glyph 'dollaroldstyle'.
Warning: missing glyph 'centoldstyle'.
Warning: missing glyph 'colonmonetary'.
Warning: missing glyph 'won'.
Warning: missing glyph 'naira'.
Warning: missing glyph 'guarani'.
Warning: missing glyph 'peso'.
Warning: missing glyph 'lira'.
Warning: missing glyph 'recipe'.
Warning: missing glyph 'interrobang'.
Warning: missing glyph 'interrobangdown'.
Warning: missing glyph 'dong'.
Warning: missing glyph 'pertenthousand'.
Warning: missing glyph 'pilcrow'.
Warning: missing glyph 'baht'.
Warning: missing glyph 'numero'.
Warning: missing glyph 'discount'.
Warning: missing glyph 'estimated'.
Warning: missing glyph 'openbullet'.
Warning: missing glyph 'servicemark'.
Warning: missing glyph 'quillbracketleft'.
Warning: missing glyph 'quillbracketright'.
Warning: missing glyph 'copyleft'.
Warning: missing glyph 'circledP'.
Warning: missing glyph 'referencemark'.
Warning: missing glyph 'radical'.
Warning: missing glyph 'euro'.

```

While this may seem like a long list, it is not unusual when installing fonts not specifically designed for TeX. You will get the most common symbols such as currency signs and other frequently used symbols, and chances are that you are not going to miss the lacking ones. If you want to learn more about these encodings, simply run `fontinst`'s encoding vectors through `latex` to get a DVI file containing a commented listing of all the glyphs:

```

latex 8r.etx
latex ot1.etx
latex t1.etx
latex tsl.etx

```

After `fontinst` is finished, we run all property list files (`p1`) through `pltotf` to create TeX font metric files (`tfm`) and all virtual property list files (`vp1`) files through `vptovf` to create virtual fonts (`vf`). When using the Bash shell, this can be done as follows:

```

for file in *.p1; do pltotf $file; done
for file in *.vp1; do vptovf $file; done

```

The generation of TeX font metrics, virtual fonts, and font definition files is now complete.

1.3 Installing the files

The TeX distribution supports a total of three TeX trees: a global one, a local one, and a user tree. The global tree is usually maintained by package man-

agement software. The local tree is for everything that is not part of the TeX distribution but should be available system-wide. The user tree is for private files of individual users on the system.

Fonts and everything related to them should go in the local tree if you have administrative access on the system. Putting them in the global tree is a bad idea because they might get overwritten when you update TeX; putting them in a private one will restrict access to them to a single user which is probably not what you want if you have administrative access. It is a good idea to define the variable `$TEXMF` (all trees) in a way that references `$TEXMFLOCAL` (the local tree) before `$TEXMFMAIN` (the global tree). This will allow you to install newer versions of selected packages in the local tree without updating the whole install. I recommend defining `$TEXMF` as follows in `texmf.cnf`:

```
TEXMF = {$HOMETEXMF,!!$TEXMFLOCAL,!!$TEXMFMAIN}
```

This will give you two levels on top of the global install: your local extensions will be preferred over files in the global tree and can in turn be overridden by individual users who put files in their private tree (`$HOMETEXMF`). These settings should go into the global configuration file for the kpathsea search library, `texmf.cfg`. For the rest of this section we will assume that we are installing the fonts in the local tree and that its top directory is `/usr/local/share/texmf`. The relevant branches of the local tree are as follows:

```
/usr/local/share/texmf/
/usr/local/share/texmf/dvips/
/usr/local/share/texmf/dvips/config/
/usr/local/share/texmf/fonts/
/usr/local/share/texmf/fonts/afm/
/usr/local/share/texmf/fonts/afm/adobe/
/usr/local/share/texmf/fonts/afm/adobe/sabon/
/usr/local/share/texmf/fonts/tfm/
/usr/local/share/texmf/fonts/tfm/adobe/
/usr/local/share/texmf/fonts/tfm/adobe/sabon/
/usr/local/share/texmf/fonts/type1/
/usr/local/share/texmf/fonts/type1/adobe/
/usr/local/share/texmf/fonts/type1/adobe/sabon/
/usr/local/share/texmf/fonts/vf/
/usr/local/share/texmf/fonts/vf/adobe/
/usr/local/share/texmf/fonts/vf/adobe/sabon/
/usr/local/share/texmf/pdftex/
/usr/local/share/texmf/pdftex/config/
/usr/local/share/texmf/tex/
/usr/local/share/texmf/tex/latex/
/usr/local/share/texmf/tex/latex/adobe/
/usr/local/share/texmf/tex/latex/adobe/sabon/
/usr/local/share/texmf/xdvi/
/usr/local/share/texmf/xdvi/config/
```

The main components of this directory structure are defined by the TeX Directory Structure (TDS),¹ another standard introduced to cope with the large

¹ <http://www.tug.org/tds/>

number of files that make up a typical Tex system. The appropriate locations for the different file types should be more or less obvious. The `fonts/` branch has subdirectories for Ascii font metrics (`afm/`), Tex font metrics (`tfm/`), Type 1 fonts (`type1/`), and virtual fonts (`vf/`). It is customary to create subdirectories for the foundry and for each font family. You can take the names of these subdirectories from the Fontname scheme as well, although this is not a requirement. The standard directory name for the foundry is given in the file `supplier.map`, the standard name for the typeface in `typeface.map`. Here are the relevant lines from both files for Sabon:

```
p adobe      @r{Adobe (@samp{p} for PostScript)}
sb sabon     Sabon b:ClassicalGaramondBT
```

The font description files (`fd`) for Latex go in a subdirectory of `tex/latex/`. The exact location is up to you but I recommend using the `foundry/typeface` scheme as well. We do not need the directories `dvips/`, `pdftex/`, and `xdvi/` at this point, but we are going to use them later. Now we create all directories and copy the files into the local tree as follows:

```
cp *.afm /usr/local/share/texmf/fonts/afm/adobe/sabon/
cp *.tfm /usr/local/share/texmf/fonts/tfm/adobe/sabon/
cp *.pfb /usr/local/share/texmf/fonts/type1/adobe/sabon/
cp *.vf  /usr/local/share/texmf/fonts/vf/adobe/sabon/
cp *.fd  /usr/local/share/texmf/tex/latex/adobe/sabon/
```

All files left in the working directory will not be used any more and may be deleted.

1.4 Creating map files

All the files that Tex and Latex need in order to use Sabon are now available. At this point we could create a perfectly valid `DVI` file with the right amount of blank space for every glyph – but we would not see a single glyph when looking at a `DVI` preview. Note that Tex itself is completely indifferent to the actual font files. It will only use the metrics in the `tfm` files without accessing the glyph outlines. Rendering or embedding fonts is at the responsibility of the application which displays the `DVI` file or processes it further in order to generate Postscript. `pdftex` is a special case because it combines the roles of Tex and of a `PDF` driver. All of these applications need to know which fonts to use. This information is provided in ‘map’ files which map font metrics to font outlines. We will deal with the three most popular applications, the Postscript driver `dvips`, the `DVI` viewer `xdvi`, and `pdftex`. All of them need to be provided with a suitable map file. For `dvips`, the syntax of this file is explained in detail in the `dvips` manual.¹ For `pdftex`, it is explained in the `pdftex` manual, and for `xdvi` in the documentation that comes with the source distribution. Fortunately, `xdvi` and `pdftex` are capable of reading `dvips`’s map files to a certain

¹ <http://www.radicaleye.com/dvipsman/>

extent. If written with a little bit of care, dvips, pdftex, and xdvi can share the same map file. This section will explain how to do that.

Let's take a look at the first line of what will become `psb.map`, our map file for Sabon. The first column indicates the name of the raw Tex font without any file extension:

```
psbr8r Sabon-Roman "TeXBase1Encoding ReEncodeFont" <8r.enc <psbr8a.pfb
```

Since the `\latinfamily` macro reencodes all regular text fonts from Adobe Standard encoding (Fontname code 8a) to Tex Base 1 (8r) when creating metric files for Tex, it corresponds to the name of the `pfb` file with encoding 8r instead of 8a. In this case, `psbr8a.pfb` becomes `psbr8r`.

```
psbr8r Sabon-Roman "TeXBase1Encoding ReEncodeFont" <8r.enc <psbr8a.pfb
```

The second column is the Postscript name of the font. Do not try to guess the right name or copy it from some map file you found somewhere on the web some time ago. If your font is included in one of the foundry-specific lists of the Fontname scheme, the Postscript name is given in the second column of the respective table. If it is not or if you are in doubt, the Postscript name should be taken from the header of the `afm` file for every font. Here are a few lines from `psbr8a.afm`:

```
StartFontMetrics 2.0
Comment Copyright (c) 1989 Adobe Systems Incorporated. All Rights Reserved.
Comment Creation Date:Fri Mar 10 16:47:51 PST 1989
FontName Sabon-Roman
FullName 12 Sabon* Roman 05232
FamilyName Sabon
EncodingScheme AdobeStandardEncoding
```

The relevant part is the line starting with “FontName” – the Postscript name of this font is “Sabon-Roman.” For each font, we copy this name verbatim to `psb.map`.

```
psbr8r Sabon-Roman "TeXBase1Encoding ReEncodeFont" <8r.enc <psbr8a.pfb
```

The third column of our map file is a reencoding instruction. As mentioned above, the `\latinfamily` macro reencodes all fonts from Adobe Standard encoding to Tex Base 1 when creating metric files for Tex. This affects the metrics only, which are defined in the `tfm` files generated by `fontinst`, while the glyph outlines as defined in the `pfb` file still use the font's native encoding. Therefore, we add a reencoding directive to the map file that will instruct all applications dealing with the actual glyph outlines to reencode them accordingly.

```
psbr8r Sabon-Roman "TeXBase1Encoding ReEncodeFont" <8r.enc <psbr8a.pfb
```

Finally, the last column contains a list of files that dvips will embed in the Postscript file. In this case, we need the Postscript encoding vector `8r.enc` for Tex Base 1 encoding and the `pfb` file, since we want the fonts to be embedded in the Postscript file. Now the map file for our basic Sabon set looks like this:

```
psbr8r Sabon-Roman "TeXBase1Encoding ReEncodeFont" <8r.enc <psbr8a.pfb
psbri8r Sabon-Italic "TeXBase1Encoding ReEncodeFont" <8r.enc <psbri8a.pfb
psbb8r Sabon-Bold "TeXBase1Encoding ReEncodeFont" <8r.enc <psbb8a.pfb
psbbi8r Sabon-BoldItalic "TeXBase1Encoding ReEncodeFont" <8r.enc <psbbi8a.pfb
```

In addition to that, we need to tell dvips about the slanted versions of all upright fonts which `\latinfamily` creates by default. We copy the lines for Sabon-Roman and Sabon-Bold and insert `o`, the Fontname code for slanted fonts, after the weight code of the TeX font name; `psbr8r` becomes `psbro8r` and `psbb8r` is changed to `psbbo8r`:

```
psbro8r Sabon-Roman "0.167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc <psbr8a.pfb
psbbo8r Sabon-Bold "0.167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc <psbb8a.pfb
```

Note that the name of the `pfb` file does *not* change. We also add a “`SlantFont`” instruction to the third column. By default, `\latinfamily` uses a slant factor of 0.167 when creating the modified metrics and our map file has to indicate this accordingly. Our complete map file looks like this:

```
psbr8r Sabon-Roman "TeXBase1Encoding ReEncodeFont" <8r.enc <psbr8a.pfb
psbri8r Sabon-Italic "TeXBase1Encoding ReEncodeFont" <8r.enc <psbri8a.pfb
psbb8r Sabon-Bold "TeXBase1Encoding ReEncodeFont" <8r.enc <psbb8a.pfb
psbbi8r Sabon-BoldItalic "TeXBase1Encoding ReEncodeFont" <8r.enc <psbbi8a.pfb
psbro8r Sabon-Roman "0.167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc <psbr8a.pfb
psbbo8r Sabon-Bold "0.167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc <psbb8a.pfb
```

The format suggested here is suitable for `xdvi`, `dvips`, and `pdftex`. We copy `psb.map` to the branch `dvips/config/` in the local TeX tree. In order to configure `dvips`, we locate the default configuration file of `dvips` (`config.ps`) in the main TeX tree and copy it to the same location. If the search order for all TeX trees is set up as suggested above, this local copy will now be picked up instead of the global one. We open this file in a text editor, locate the section for map files (lines defining map files begin with a lowercase “`p`”), and add the new map file so that the updated section looks as follows:

```
% Standard map file provided by default
p +psfonts.map
% New map file for Sabon
p +psb.map
```

The procedure for `pdftex` is similar: the configuration file is called `pdftex.cfg` and map files are marked with the string “`map`” at the beginning of the line. After copying the file to the branch `pdftex/config` of the local tree and updating it, the relevant section should look similar to the following example:

```
% Standard map file provided by default
map +pdftex.map
% New map file for Sabon
map +psb.map
```

We repeat this step one more time for `xdvi`. The configuration file for `xdvi` is called `xdvi.cfg`, the local branch is `xdvi/config` and lines indicating a map file begin with “`dvipsmap`”:


```
% Map files provided by default
dvipsmap ps2pk.map
dvipsmap ...
% New map file for Sabon
dvipsmap psb.map
```

In addition to that, we have to make sure that an encoding definition for TeX Base 1 encoding is provided as well. The configuration file for xdvi should contain the following line:

```
% Tag      Suffix  Encoding name      Encoding file
enc       8r      TeXBase1Encoding  8r.enc
```

The installation is now finished. Do not forget to update the file hash tables by running `texhash` or an equivalent command!

1.5 Using the fonts

Everything you need to know about using the fonts can be found in the LaTeX font selection guide.¹ The second chapter of this guide documents the standard NFSS commands used to switch fonts under LaTeX. Let's take a look at some examples. To select Sabon at any point in a LaTeX file, we use a command like:

```
\fontfamily{psb}\selectfont
```

Sabon provides two weights which are readily available using compact font selection macros like `\textbf` and `\bfseries`. Larger font families may offer more than two weights. To select a particular weight, we use the `\fontseries` command in conjunction with the NFSS series codes defined during the installation of the font family. Please refer to the code table on page 84 of this guide for a list of the most common NFSS codes. To select the semibold (`sb`) weight for example, we would use the following construct:

```
\fontseries{sb}\selectfont
```

Compact font switching macros such as `\mdseries` and `\bfseries` do not switch to a fixed NFSS font series, they use `\mddefault` and `\bfdefault` for the regular and bold weight respectively. If we want to use semibold as the default bold weight, for example, we simply redefine `\bfdefault` accordingly:

```
\renewcommand*\bfdefault{sb}
```

In order to use Sabon as the default roman typeface for the whole document, we redefine `\rmdefault` in the preamble:

```
\renewcommand*\rmdefault{psb}
```

It is much more convenient to put the initialization of the font family into a dedicated style file (`sty`), though. Our file `sabon.sty` might look like this:

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{sabon}[2002/04/17 v1.0 Adobe Sabon]
```

¹ <http://www.ctan.org/tex-archive/macros/latex/doc/fntguide.pdf>

```

3 \RequirePackage[T1]{fontenc}
4 \RequirePackage{textcomp}
5 \renewcommand*\rmdefault}{psb}
6 \endinput

```

Essentially, we redefine `\rmdefault` in order to use Sabon as the default roman typeface for the whole document. In addition to that, we load the `fontenc` package and switch to `T1` encoding, which is more appropriate for Postscript fonts than the `OT1` encoding used by default. We also load the `textcomp` package which provides a user interface for the symbols found in `T1` encoding. This will allow us to access symbols such as ‘copyright’ or ‘registered’. If the `textcomp` package is used in conjunction with `inputenc`, it is even possible to enter most of these symbols directly in a Latex file.

There is one thing we have to keep in mind when switching to `T1` encoding. The default encoding is a global setting that applies to all text fonts used in a Latex file, unless the encoding is reset explicitly using the `\fontencoding` macro `\fontencoding`. It will affect the font family defined as `\rmdefault`, but also the families set up as `\sfdefault` and `\ttdefault`. By default, these are Computer Modern Sans Serif (`cms`) and Computer Modern Typewriter (`cmtt`). Using these fonts in conjunction with `T1` encoding will pose some problems most European Tex users are already well familiar with. It is perfectly possible, provided that we use a suitable version of the Computer Modern fonts. Choosing a suitable version, however, can be quite difficult. We will discuss some typical issues related to that in the following section. Alternatively, we could use other `T1` encoded sans serif and typewriter typefaces available in Postscript format. For example, here is an enhanced version of `sabon.sty` using Helvetica (`phv`) and Courier (`pcr`):

```

1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{sabon}[2002/04/17 v1.0 Adobe Sabon with PS fonts]
3 \RequirePackage[T1]{fontenc}
4 \RequirePackage{textcomp}
5 \renewcommand*\rmdefault}{psb}
6 \renewcommand*\sfdefault}{phv}
7 \renewcommand*\ttdefault}{pcr}
8 \endinput

```

This setup is certainly not the most fortunate one in terms of typography, but it should be safe from a technical perspective. Helvetica and Courier are part of the Postscript base fonts built into every Level 2 Postscript device. Most Tex distributions do not ship with the original versions of these fonts but they provide suitable replacements for them. For our setup of Sabon, the next section is only relevant if you want to use Computer Modern Sans Serif and Computer Modern Typewriter in conjunction with Sabon. If you deploy different `T1` encoded sans serif and typewriter typefaces, which are available in Postscript format, all you need to do is redefine `\sffamily` and `\ttfamily` in `sabon.sty` or in the preamble of the respective Latex file as shown above for Helvetica and Courier.

1.6 Computer Modern and T1 encoding

The Computer Modern fonts designed for T1 and TS1 encoding are called EC and TC fonts respectively, together known as European Computer Modern. When switching to T1 encoding, we implicitly switch to these fonts. Note that European Computer Modern, while being derived from Donald Knuth's original Computer Modern typefaces, is not simply a T1 encoded drop-in replacement. Over the years it has evolved into an independent typeface. The additional fonts created for the European Computer Modern family have been subject to debate based on their design. Some of them are considered to be typographically inferior to the original designs. From a technical perspective, the problem with the European Computer Modern fonts is that, historically, they have been available in Metafont format only. This implies that Postscript and PDF files will contain bitmap representations of these fonts when we switch to T1 encoding. Bitmap fonts, however, have a fixed resolution and so are not independent of the output device. They are not suitable for on-screen display and a major inconvenience for every print shop, if they are tolerated at all.

Donald Knuth had designed the Computer Modern fonts in Metafont format and with OT1 encoding in mind. Blue Sky Research and Y&Y developed Postscript versions of these fonts later, which were donated to the public in 1997 and have been shipping with most Tex distributions ever since. While these fonts work fine for Postscript and PDF files, they are not suitable for tasks requiring letters not found in the English alphabet because their glyph base is still restricted to OT1 encoding. Jörg Knappen's European Computer Modern fonts address this issue by providing a more comprehensive set of glyphs, but they have in turn been subject to the limitations of Metafont. In the following, I will briefly introduce several solutions which try to address these problems. Most of them are trade-offs in one way or another. Tables 1 and 2 try to provide an overview of the major design variations over the Computer Modern theme along with their implementations. The tables are by no means exhaustive, there are even more fonts derived from the original Computer Modern typefaces.

To work around the hyphenation problem of OT1 encoding while sticking to the original Computer Modern fonts, there is a choice of two packages on CTAN which provide T1 encoded virtual fonts based on the original Computer Modern family of fonts: the AE¹ and the ZE² fonts. The AE fonts are built on top of Computer Modern exclusively, but unfortunately they lack almost a dozen T1 characters including the French double and single guillemets, which makes their default setup unsuitable for all French and a lot of German texts. For Computer Modern Typewriter, the situation is even worse. There is a supplemental package called `aecompl` which adds Metafont versions of the missing characters, but that again brings up the problem we were trying to avoid in the first

¹ <http://www.ctan.org/tex-archive/fonts/ae/>

² <http://www.ctan.org/tex-archive/fonts/zefonts/>

TYPEFACE	FONTS	
	NAME	FORMAT
Computer Modern	CM	Metafont
	CM, Blue Sky	Postscript
	CM, Bakoma	Postscript
	AE	virtual fonts
	ZE	virtual fonts
European Computer Modern	EC & TC	Metafont
	EC & TC, Micropress	Postscript
	Tt2001	Postscript
	CM-super	Postscript
Latin Modern	LM	Postscript
European Modern	EM	Postscript

TABLE 1: Computer Modern, fonts and formats

place. A different complement called `aeguill`¹ at least adds Postscript versions of the guillemets. An enhanced version of `sabon.sty` might then look like this:

```

1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{sabon}[2002/04/17 v1.0 Adobe Sabon with AE]
3 \RequirePackage[T1]{fontenc}
4 \RequirePackage{textcomp}
5 \RequirePackage{ae}
6 \RequirePackage{aeguill}
7 \renewcommand*{\rmdefault}{psb}
8 \endinput

```

The `ZE` fonts take a different approach to work around this problem: the missing characters are taken from standard Postscript fonts such as Times and Helvetica. This means that there will be some typographical inconsistencies, but we are safe from a technical point of view. While the `AE` fonts and the corresponding supplemental packages ship with most TeX distributions, you might need to download the `ZE` fonts from CTAN. When using the `ZE` fonts, our enhanced version of `sabon.sty` would look like this:

```

1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{sabon}[2002/04/17 v1.0 Adobe Sabon with ZE]
3 \RequirePackage[T1]{fontenc}
4 \RequirePackage{textcomp}
5 \RequirePackage{zefonts}
6 \renewcommand*{\rmdefault}{psb}
7 \endinput

```

There is a more robust solution you might be interested in if you require `T1` encoded Computer Modern fonts. Free Postscript versions of the European Computer Modern fonts have been made available, although they might not have made their way into every TeX distribution yet. As mentioned before, one prob-

¹ <http://www.ctan.org/tex-archive/macros/latex/contrib/supported/aeguill/>

FONTS	ENCODING	
	NATIVE	SUPPORTED
CM	OT1	OT1
CM, Blue Sky	OT1	OT1
CM, Bakoma	font specific	OT1
AE	OT1	T1, with composite glyphs
ZE	OT1	T1, with composite glyphs
EC, TC	T1, TS1	T1, TS1
CM-super	8a	T1, TS1, T2A, T2B, T2C, X2
LM	font specific	T1, TS1, LY1, QX1

TABLE 2: Computer Modern, fonts and encodings

lem with OT1 encoded fonts is that they rely on composite glyphs which break searching for words containing accented letters in PDF files. Both the AE and the ZE fonts, although they enable Tex to hyphenate words containing accented letters properly, still suffer from this particular problem as they are based on OT1 encoded fonts internally. It is highly advisable to switch to a real T1 version of the Computer Modern fonts in Postscript format. Such fonts are included in two independent packages: Péter Szabó's Tt2001¹ as well as Vladimir Volovich's more recent CM-super² package. Both packages include Postscript fonts which are traced and post-processed conversions of their Metafont counterparts.

Unless you know that a specific font you need is provided by the Tt2001 package only, go with the more advanced CM-super package which will bring you as close to a real solution as you can possibly get when using free versions of the European Computer Modern fonts. Note, however, that it is a rather large download. Since it includes a huge number of fonts, the compressed package is about 64 MB in size. The CM-super fonts use Adobe Standard as their native encoding, but the glyph set provided by these fonts includes Cyrillic letters as well. In addition to T1 and TS1, CM-super supports the Cyrillic encodings T2A, T2B, T2C, and X2. See the package documentation for installation instructions and answers to the most frequently asked questions. Here is a version of our style file for use in conjunction with CM-super:

```

1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{sabon}[2002/04/17 v1.0 Adobe Sabon with CM-Super]
3 \RequirePackage[T1]{fontenc}
4 \RequirePackage{textcomp}
5 \RequirePackage{typelec}
6 \renewcommand*{\rmdefault}{psb}
7 \endinput

```

Recently, yet another new implementation of Computer Modern has been re-

1 <http://www.ctan.org/tex-archive/fonts/ps-type1/ec/>
2 <http://www.ctan.org/tex-archive/fonts/ps-type1/cm-super/>

leased to the public, the promising Latin Modern fonts created by Bogusław Jackowski and Janusz M. Nowacki. Unlike Tt2001 and CM-super, Latin Modern is derived from the original Computer Modern designs and augmented with accented letters as well as other glyphs missing from the very restricted glyph base of the original fonts. While the Latin Modern fonts are younger than European Computer Modern, they are a parallel development from a systematic perspective. Consequently, they are not affected by the controversial design decisions underlying certain parts of the European Computer Modern family of fonts. They use a font specific encoding by default and feature a glyph base suitable for T1, TS1, LY1 as well as the Polish encoding QX1. Even though these fonts are still under development, they are already perfectly usable as of this writing. Here is yet another iteration of our style file for Sabon, combined with Latin Modern for the sans serif and typewriter families:

```

1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{sabon}[2003/07/27 v1.0 Adobe Sabon with LM]
3 \RequirePackage[T1]{fontenc}
4 \RequirePackage{textcomp}
5 \RequirePackage{lmodern}
6 \renewcommand*{\rmdefault}{psb}
7 \endinput

```

Apart from these free fonts, there are also commercial offerings from Y&Y¹ and Micropress.² Judging by the vendors' websites, Micropress offers Postscript versions of European Computer Modern while the European Modern fonts by Y&Y are augmented Postscript versions of the original Computer Modern typefaces. Please refer to the respective website for details and pricing. Since I have never used any of these fonts, I cannot comment on their quality or on any possible shortcomings.

¹ <http://www.yandy.com/em.htm>

² <http://www.micropress-inc.com/fonts/ecfonts/ecmain.htm>

TUTORIAL II

STANDARD FONT SETS

While the `\latinfamily` shorthand is very convenient, it is not capable of coping with complex installation scenarios. Sooner or later you will probably have more specific requirements or simply desire more control over the basics. This will require using lower-level `fontinst` commands in most cases.

II.1 The `fontinst` file

In this tutorial, we will essentially repeat the scenario discussed in the previous one. This time, however, we will employ lower-level commands. The verbose file introduced here will also serve as a template for subsequent tutorials.

```
1 \input fontinst.sty
2 \substitutesilent{bx}{b}
```

After loading `fontinst` we set up an alias that will suppress a warning when the respective font is substituted. Why would we want to set up this particular alias? Note that `bx` is the NFSS code of the ‘bold extended’ series. The LaTeX macros `\textbf` and `\bfseries` do not switch to a fixed series, they use `\bfdefault` instead which is set to `bx` by default. As long as you are using the Computer Modern fonts this is fine since they actually include bold extended fonts. For font families which do not, however, using these macros would result in a warning. To avoid that, you would need to redefine `\bfdefault` to a suitable weight. The problem here is that `\bfdefault` is a global setting applying to all of LaTeX’s font families (`\rmdefault`, `\sfdefault`, and `\ttdefault`), but it is not safe to assume that all of them will offer the same weights. To avoid any need to redefine `\bfdefault` unless we really want to, we set up an alias so that every request for ‘bold extended’ (`bx`) is substituted by ‘bold’ (`b`).¹ Unless bold extended fonts are available, simply think of `bx` as the default bold weight.

The standard weight is selected by LaTeX in a similar way. The relevant macro is called `\mddefault` and defaults to `m`. Make sure that the NFSS series `m` is always defined, either mapped to actual fonts or as a substitution. In this case our font family provides regular-weight fonts so we will simply use them for the `m` series. Some font families, however, are based on the main weights ‘light’ and ‘demibold’ instead of ‘regular’ and ‘bold’. In this case, we would either just map these weights to the `m` and `b` series directly or use the proper NFSS series codes (`l` and `db`) plus the following substitutions:

```
\substitutesilent{m}{l}
\substitutesilent{bx}{db}
```

¹ This is a default substitution that `fontinst` will always silently include. We could omit line 2 here, but if semibold fonts are available you might prefer using those as a substitute for `bx`.

Again, think of `m` as the default weight if regular-weight fonts are not available. Every font family should provide mappings for the NFSS series `m` and `bx` in the font definition file. If fonts matching these series exactly are not available, use substitutions to ensure that the defaults for `\mddefault` and `\bfdefault` will work without user intervention. Since `\mddefault` and `\bfdefault` are overall settings applying to all of LaTeX's families, redefining them explicitly may cause problems. Doing so should be an option, not a requirement.

```
3 \substitutesilent{sc}{n}
```

We also add a substitution for the `sc` shape, which will in fact be used by the `TS1` encoded families only. Since `TS1` contains symbols and figures, we do not need an additional small caps font for this encoding as it would be identical to the upright variant anyway. However, to ensure that all text commands of the `textcomp` package will always work, even if the active NFSS shape is `sc`, we set up this shape substitution.

```
4 \setint{smallcapsscale}{800}
```

The basic Sabon set we are dealing with offers upright and italic fonts but no optical small caps. As a substitute, `fontinst` is capable of transparently generating so-called ‘mechanical’ or ‘faked’ small caps – as opposed to ‘optical’ or ‘real’ small caps which are actual glyphs found in a dedicated small caps font. Mechanical small caps are generated by taking the tall caps of the font and scaling them by a certain factor: 1000 means full size, 800 means 0.8. Since Type 1 fonts scale linearly, scaling down tall caps implies that they will appear lighter than the corresponding lowercase glyphs, thus disturbing the color of the page. However, if they are too tall they do not mix well with the lowercase alphabet.

Optical small caps match the ‘x-height’ of the font. This is the height of the lowercase alphabet without ascenders and descenders. They blend in seamlessly with lowercase and mixed case text. Depending on the typeface, this usually corresponds to a value in the range of 650–750. If you scale down tall caps so that they match the x-height of the font, they will appear too light in running text. Finding a suitable value for this is obviously a trade-off. We are going to use `fontinst`'s default setting of 800 here but you might want to experiment with a value in the range of 750–800. For serious applications of small caps we would need optical small caps, provided in a dedicated small caps or in an expert font. For details on small caps and expert sets, please refer to tutorial III and V respectively.

```
5 \setint{slant}{167}
```

The integer variable `smallcapsscale` is a predefined variable used by `fontinst`'s encoding vectors. We could use it in conjunction with `\latinfamily` as well. The variable `slant` is specific to our `fontinst` file. We define it for convenience so that we can set the slant factor for all subsequent font transformations globally. The slant factor defines how much the glyphs slope to the right. It is a real

number equivalent to the tangent of the slant angle. Fontinst represents this number as an integer though, so we have to multiply the tangent by 1000. The value 167 ($\sim 9.5^\circ$) is a reasonable default. Any value significantly greater than 176 ($\sim 10^\circ$) is usually too much.¹

```
6 \transformfont{psbr8r}{\reencodefont{8r}{\fromafm{psbr8a}}}
7 \transformfont{psbri8r}{\reencodefont{8r}{\fromafm{psbri8a}}}
8 \transformfont{psbb8r}{\reencodefont{8r}{\fromafm{psbb8a}}}
9 \transformfont{psbbi8r}{\reencodefont{8r}{\fromafm{psbbi8a}}}
```

We start off with some basic font transformations: all fonts are reencoded from Adobe Standard (Fontname code 8a) to Tex Base 1 encoding (8r). Please refer to the fontinst manual for an explanation of the syntax of the individual commands used here and in the following.

```
10 \transformfont{psbro8r}{\slantfont{\int{slant}}\reencodefont{8r}{\fromafm{psbr8a}}}
11 \transformfont{psbbo8r}{\slantfont{\int{slant}}\reencodefont{8r}{\fromafm{psbb8a}}}
```

Like the `\latinfamily` shorthand, our fontinst file should create slanted fonts as well. These need to be reencoded and, well, slanted. We are using the `slant` variable defined in line 5 to set the slant factor. The raw, Tex Base 1 encoded fonts are now prepared for the generation of virtual fonts.

```
12 \installfonts
13 \installfamily{T1}{psb}{}
14 \installfamily{TS1}{psb}{}

```

The installation of a font family is enclosed in an environment which we open in line 12 and close later in line 29. First of all, the font family we are about to install has to be declared: we have Adobe Sabon and we are going to install it in `T1` encoding (Fontname code 8t) as well as in `TS1` (8c). The third argument to `\installfamily` corresponds to the second one of the `\latinfamily` command: it is used to include code in the font definition file that will be read by LaTeX whenever the font is selected. `T1` will serve as our base encoding in LaTeX's text mode later. It is complemented by `TS1` which provides additional glyphs such as currency signs and other frequently used symbols. The `\latinfamily` command also provides `OT1` (7t) and Tex Base 1 encoded fonts. We will omit both encodings here as we do not need them. While raw Tex Base 1 encoded fonts (8r) form the basis of all virtual fonts, they are usually not deployed as such on the Tex level, and the `OT1` encoding is not suitable for Postscript fonts anyway. We will therefore deliberately ignore it and focus on `T1` and `TS1` exclusively.

```
15 \installfont{psbr8t}{psbr8r,latin}{t1}{T1}{psb}{m}{n}{}

```

¹ I suggest you do not bother trying to match the slope of the italic fonts when creating a slanted variant of a roman font. This will usually not work for typefaces with true italics because the latter are an independent design.

To create the individual virtual fonts, we use fontinst's `\installfont` command. The first argument to `\installfont` is the virtual font we are going to create, the second one is a list of files used to build this font. These can be `afm`, `mtx`, or `p1` files, their extension is omitted. If multiple fonts are provided, `\installfont` does not overwrite any encoding positions when reading in additional files, it simply fills vacant slots if it finds suitable glyphs in the next font. The metric file `latin.mtx` is an auxiliary file provided by fontinst which should always be read when creating `OT1` or `T1` encoded text fonts. The third argument is the file name of an encoding vector without the file extension, in this case `t1.etx`. The remaining arguments are written verbatim to the font definition file and declare the respective font in a format that the LaTeX font selection scheme (NFSS) can process: `T1` encoding, Adobe Sabon¹, medium², normal (that is, upright or roman). The last argument is only relevant if fonts with different design sizes are available. It is empty for linearly scaled fonts.

```
16 \installfont{psbrc8t}{psbr8r,latin}{t1c}{T1}{psb}{m}{sc}{}

```

The small caps font is slightly different. Since we do not have any Type 1 font containing optical small caps we need to ‘fake’ them by scaling the uppercase alphabet and putting the scaled glyphs in the encoding positions of the lowercase alphabet. Fortunately, we do not have to deal with the actual low-level glyph scaling. We simply load `t1c.etx`, a special encoding vector which will take care of that, using the value of `smallcapsscale` as the scale factor.

```
17 \installfont{psbro8t}{psbro8r,latin}{t1}{T1}{psb}{m}{s1}{}
18 \installfont{psbri8t}{psbri8r,latin}{t1}{T1}{psb}{m}{it}{}

```

Since the slanting was already performed on the raw fonts, the virtual slanted and the italic fonts are handled just like the upright ones. Now all regular fonts are done and we can repeat this part (15–18) for the bold fonts:

```
19 \installfont{psbb8t}{psbb8r,latin}{t1}{T1}{psb}{b}{n}{}
20 \installfont{psbbc8t}{psbb8r,latin}{t1c}{T1}{psb}{b}{sc}{}
21 \installfont{psbbo8t}{psbbo8r,latin}{t1}{T1}{psb}{b}{s1}{}
22 \installfont{psbbi8t}{psbbi8r,latin}{t1}{T1}{psb}{b}{it}{}

```

After that, we add virtual fonts for `TS1` encoding:

```
23 \installfont{psbr8c}{psbr8r,textcomp}{ts1}{TS1}{psb}{m}{n}{}

```

Like `latin.mtx`, `textcomp.mtx` is an auxiliary metric file provided by fontinst. It should always be added when creating `TS1` encoded fonts for the `textcomp` package. The third argument, the encoding vector, refers to `ts1.etx` in this case. As `TS1` encoding is for symbols only and we did set up a shape substitution, we do not need a `TS1` encoded small caps font. Slanted and italic fonts are handled like the upright one:

- 1 LaTeX does not really care about the name of the font or the foundry. This argument simply defines the code that identifies the font within the NFSS.
- 2 In fact, the more appropriate name would be *regular* because *medium* is a moderate bold weight with the NFSS code `mb`.

```

24 \installfont{psbro8c}{psbro8r,textcomp}{ts1}{TS1}{psb}{m}{s1}{ }
25 \installfont{psbri8c}{psbri8r,textcomp}{ts1}{TS1}{psb}{m}{it}{ }

```

We repeat 23–25 for the bold fonts:

```

26 \installfont{psbb8c}{psbb8r,textcomp}{ts1}{TS1}{psb}{b}{n}{ }
27 \installfont{psbbo8c}{psbbo8r,textcomp}{ts1}{TS1}{psb}{b}{s1}{ }
28 \installfont{psbbi8c}{psbbi8r,textcomp}{ts1}{TS1}{psb}{b}{it}{ }

```

Finally, we close the install environment and terminate:

```

29 \endinstallfonts
30 \bye

```

II.2 The *latinfamily* macro revisited

Note that our `fontinst` file is not strictly equivalent to the `\latinfamily` macro but rather stripped down to the most useful parts with respect to typical Postscript fonts. Essentially, we did not create any font description files for the raw Tex Base 1 encoded fonts and we dropped OT1 encoding. If you are curious, you should be able to reconstruct all the steps taken by `\latinfamily` when looking at the log file created by `fontinst` while keeping our file in mind. Here are the relevant lines from the log file after running `\latinfamily` on the basic Sabon set. Only lines beginning with “INFO> run” are relevant in this context as they indicate lower-level macros used by `\latinfamily`:

```

INFO> run \transformfont <psbr8r> from <psbr8a>
INFO> run \installrawfont <psbr8r><psbr8r,8r><8r><8r><psb><m><n>
INFO> run \installfont <psbr7t><psbr8r,latin><OT1><OT1><psb><m><n>
INFO> run \installfont <psbr8t><psbr8r,latin><T1><T1><psb><m><n>
INFO> run \installfont <psbr8c><psbr8r,textcomp><TS1><TS1><psb><m><n>
INFO> run \installfont <psbrc7t><psbr8r,latin><OT1c><OT1><psb><m><sc>
INFO> run \installfont <psbrc8t><psbr8r,latin><T1c><T1><psb><m><sc>
INFO> run \transformfont <psbro8r> from <psbr8r> (faking oblique)
INFO> run \installrawfont <psbro8r><psbro8r,8r><8r><8r><psb><m><s1>
INFO> run \installfont <psbro7t><psbro8r,latin><OT1><OT1><psb><m><s1>
INFO> run \installfont <psbro8t><psbro8r,latin><T1><T1><psb><m><s1>
INFO> run \installfont <psbro8c><psbro8r,textcomp><TS1><TS1><psb><m><s1>
INFO> run \transformfont <psbri8r> from <psbri8a>
INFO> run \installrawfont <psbri8r><psbri8r,8r><8r><8r><psb><m><it>
INFO> run \installfont <psbri7t><psbri8r,latin><OT1i><OT1><psb><m><it>
INFO> run \installfont <psbri8t><psbri8r,latin><T1i><T1><psb><m><it>
INFO> run \installfont <psbri8c><psbri8r,textcomp><TS1i><TS1><psb><m><it>
INFO> run \transformfont <psbb8r> from <psbb8a>
INFO> run \installrawfont <psbb8r><psbb8r,8r><8r><8r><psb><b><n>
INFO> run \installfont <psbb7t><psbb8r,latin><OT1><OT1><psb><b><n>
INFO> run \installfont <psbb8t><psbb8r,latin><T1><T1><psb><b><n>
INFO> run \installfont <psbb8c><psbb8r,textcomp><TS1><TS1><psb><b><n>
INFO> run \installfont <psbbc7t><psbb8r,latin><OT1c><OT1><psb><b><sc>
INFO> run \installfont <psbbc8t><psbb8r,latin><T1c><T1><psb><b><sc>
INFO> run \transformfont <psbbo8r> from <psbb8r> (faking oblique)
INFO> run \installrawfont <psbbo8r><psbbo8r,8r><8r><8r><psb><b><s1>
INFO> run \installfont <psbbo7t><psbbo8r,latin><OT1><OT1><psb><b><s1>
INFO> run \installfont <psbbo8t><psbbo8r,latin><T1><T1><psb><b><s1>
INFO> run \installfont <psbbo8c><psbbo8r,textcomp><TS1><TS1><psb><b><s1>
INFO> run \transformfont <psbbi8r> from <psbbi8a>

```

```

INFO> run \installrawfont <psbbi8r><psbbi8r,8r><8r><8r><psb><b><it>
INFO> run \installfont <psbbi7t><psbbi8r,latin><OT1i><OT1><psb><b><it>
INFO> run \installfont <psbbi8t><psbbi8r,latin><T1i><T1><psb><b><it>
INFO> run \installfont <psbbi8c><psbbi8r,textcomp><TS1i><TS1><psb><b><it>

```

This listing is a complete summary of what the `\latinfamily` macro does in this case, broken down into lower-level commands. The order of the commands differs slightly from our file, because the `\transformfont` calls are not grouped at the beginning but rather used ‘on demand’ for each shape. This difference is irrelevant from a technical point of view. `\transformfont` must obviously be called before `\installfont` or `\installrawfont` tries to use the transformed fonts, but the exact location does not matter. Since we did not create any font description files for Tex Base 1 encoding, we did not use the `\installrawfont` macro in our fontinst file. This macro does not build a virtual font but rather sets up a raw, Tex Base 1 encoded font for use under Latex.

Here are some crucial points we would have to keep in mind when writing a fontinst file that does exactly what `\latinfamily` would do: the macro `\installrawfont` is used in conjunction with `8r.mtx` instead of `latin.mtx`, the encoding file is obviously `8r.etx` in this case. Creating OT1 encoded virtual fonts requires `latin.mtx` and `ot1.etx`. You will also notice that, in addition to `ot1c.etx` and `t1c.etx`, fontinst used encoding files like `ot1i.etx` and `t1i.etx` when creating italic virtual fonts. For T1 encoding, `t1.etx` and `t1i.etx` are equivalent because `t1i.etx` reads `t1.etx` internally, hence we did not use `t1i.etx` in our fontinst file. The situation is the same with `ts1.etx` and `ts1i.etx`. For OT1 encoding, however, the difference is crucial because this encoding differs depending on the shape: the upright shape features a dollar symbol while the italic shape puts an italic pound symbol in the slot of the dollar. This is yet another idiosyncrasy of OT1.

11.3 Map files revisited

With all of that in mind, let’s now go back to the dvips map file from the first tutorial and take another look at it. The meaning of the reencoding and slanting instructions should be much clearer now:

```

psbr8r Sabon-Roman "TeXBase1Encoding ReEncodeFont" <8r.enc <psbr8a.pfb
psbri8r Sabon-Italic "TeXBase1Encoding ReEncodeFont" <8r.enc <psbri8a.pfb
psbb8r Sabon-Bold "TeXBase1Encoding ReEncodeFont" <8r.enc <psbb8a.pfb
psbbi8r Sabon-BoldItalic "TeXBase1Encoding ReEncodeFont" <8r.enc <psbbi8a.pfb
psbro8r Sabon-Roman "0.167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc <psbr8a.pfb
psbbo8r Sabon-Bold "0.167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc <psbb8a.pfb

```

Note that the T1 and TS1 encodings are used for the virtual fonts only, they are what Tex will work with. A Postscript file created by dvips, however, does not contain any virtual fonts. They will have been resolved into the raw fonts they are based on by dvips. The raw fonts used to build virtual ones were reencoded to Tex Base 1 encoding during the installation. But this reencoding step affects the font metrics only while the pfb files embedded in the Postscript code still

use Adobe Standard as their native encoding. Therefore every application reading the final file has to repeat the reencoding step for the font outlines before rendering the fonts. This is what the “ReEncodeFont” instruction is all about. Since we cannot expect every application to know about Tex Base 1 encoding, we embed the respective encoding vector (`8r.enc`) along with the fonts. Compare the first `\transformfont` command in the fontinst file to the first line of the map file:

```
\transformfont{psbr8r}          {\reencodefont{8r}          {\fromafm{psbr8a}}}
psbr8r Sabon-Roman "TeXBase1Encoding ReEncodeFont" <8r.enc <psbr8a.pfb
```

The situation is similar for the slanted fonts. The font files embedded in the Postscript file are not slanted, they are upright. Fontinst has performed the slanting for the font metrics only, it does not touch the font outlines at all. The slanting of the glyph outlines will be performed by a Postscript printer or an interpreter like Ghostscript. After resolving the virtual fonts, all that dvips does as far as the raw fonts are concerned is reading the files listed in `psb.map` and embedding them along with the “SlantFont” instruction. The transformation of the glyph outlines takes place when the Postscript code is rendered on screen or on paper. Both “ReEncodeFont” and “SlantFont” are instructions for the application finally performing the rendering. The value of the “SlantFont” instruction has to correspond to the slant factor used in the fontinst file. As mentioned above, fontinst’s representation of the slant factor is slightly different. The value used in the map file is a real number corresponding to fontinst’s (integer) slant factor divided by 1000. That’s why its precision is fixed to three decimal places. Let’s compare a line of the map file to the corresponding line of the fontinst file:

```
\transformfont{psbro8r}{\slantfont{167}\reencodefont{8r}{\fromafm{psbr8a}}}
psbro8r Sabon-Roman "0.167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc <psbr8a.pfb
```

Essentially, think of map files as a way of recording all encoding and shape modifications applied to the font metrics during the installation, so that they can be repeated for the font outlines when the final Postscript file is displayed or printed. This information is required for the raw fonts only because all the information concerning the virtual fonts is contained in the virtual font files. When using DVI or PDF as the final output format, the division of labor between the various tools involved differs since pdftex combines the roles of Tex and dvips, while DVI viewers deal with both the virtual fonts and the rendering of the font outlines on screen. The principle, however, remains the same. Therefore pdftex and xdvi require map files as well.

TUTORIAL III

OPTICAL SMALL CAPS AND HANGING FIGURES

When choosing a new typeface, bear in mind that optical small caps and hanging figures are not available for all commercial Postscript fonts. If they are available for a certain typeface, they are usually provided separately, either in a `sc & osf` or in an expert font package. We will deal with the former case in this tutorial, the latter will be discussed in tutorial v. Suppose we have acquired the Sabon `sc & osf` package to complement our base install of Sabon. This package provides four additional fonts: a regular `sc & osf`, an italic `osf`, a bold `osf`, and a bold italic `osf` font. These fonts will provide us with hanging figures for all shapes in both weights. Small caps are available for the regular weight only; we will still have to make do with mechanical small caps for the bold weight. Adobe does not include a separate regular-weight upright `osf` font. The respective figures are to be found in the small caps font instead. Our original file set looks like this:

```

sar____.afm   sai____.afm   sab____.afm   sabi____.afm
sar____.inf   sai____.inf   sab____.inf   sabi____.inf
sar____.pfb   sai____.pfb   sab____.pfb   sabi____.pfb
sar____.pfm   sai____.pfm   sab____.pfm   sabi____.pfm

sarsc____.afm   saiof____.afm   sabof____.afm   sabio____.afm
sarsc____.inf   saiof____.inf   sabof____.inf   sabio____.inf
sarsc____.pfb   saiof____.pfb   sabof____.pfb   sabio____.pfb
sarsc____.pfm   saiof____.pfm   sabof____.pfm   sabio____.pfm

```

After renaming and choosing the required files, we could start off with the following set of files:

```

psbr8a.afm   psbri8a.afm   psbb8a.afm   psbbi8a.afm
psbr8a.pfb   psbri8a.pfb   psbb8a.pfb   psbbi8a.pfb

psbrc8a.afm   psbrij8a.afm   psbbj8a.afm   psbbij8a.afm
psbrc8a.pfb   psbrij8a.pfb   psbbj8a.pfb   psbbij8a.pfb

```

But before we begin, let's take a closer look at the encoding of the fonts. We will have to deal with some peculiarities characteristic for typical `sc & osf` sets. Taking a look at `psbr8a.afm`, you will see that in Adobe Standard encoding, which is the native encoding of all fonts of the Sabon family, the figures are encoded as “zero”, “one”, “two” etc.:

```

C 48 ; WX 556 ; N zero ; B 52 -15 504 705 ;
C 49 ; WX 556 ; N one ; B 91 0 449 705 ;
C 50 ; WX 556 ; N two ; B 23 0 507 705 ;

```

Compare that to the glyph names of figures in an expert font:

```

C 48 ; WX 511 ; N zerooldstyle ; B 40 -14 480 436 ;
C 49 ; WX 328 ; N oneoldstyle ; B 35 -3 294 425 ;
C 50 ; WX 440 ; N twooldstyle ; B 44 -3 427 436 ;

```

The different glyph names are appropriate because regular Postscript fonts usually come with lining figures by default while expert fonts feature hanging (‘old style’) figures amongst other things. Now let’s take a look at `psbrc8a.afm`:

```
C 48 ; WX 556 ; N zero ; B 41 -15 515 457 ;
C 49 ; WX 556 ; N one ; B 108 0 448 442 ;
C 50 ; WX 556 ; N two ; B 72 0 512 457 ;
```

When comparing these glyph names to the actual outlines in `psbrc8a.pfb`,¹ we would see that this font in fact comes with hanging (‘old style’) figures even though the figures are labeled using the standard names. This is the case with all OSF fonts included in the `sc & OSF` package. The reason why this complicates the installation procedure will become clear when we take a look at the TeX side. In T1 encoding, for example, the figures are (essentially) encoded like this by default:

```
\setslot{zero}\endsetslot
\setslot{one}\endsetslot
\setslot{two}\endsetslot
```

While T1 encoding (essentially) references them as follows:

```
\setslot{zerooldstyle}\endsetslot
\setslot{oneoldstyle}\endsetslot
\setslot{twooldstyle}\endsetslot
```

We face a similar problem with small caps. The lowercase letters in `psbr8a.afm` are labeled like this:

```
C 97 ; WX 500 ; N a ; B 42 -15 465 457 ;
C 98 ; WX 556 ; N b ; B 46 -15 514 764 ;
C 99 ; WX 444 ; N c ; B 25 -15 419 457 ;
```

Expert fonts, which provide small caps as well but do not need to follow Adobe Standard encoding, encode small caps as follows:

```
C 97 ; WX 457 ; N Asmall ; B -15 -3 467 446 ;
C 98 ; WX 481 ; N Bsmall ; B 34 -3 437 437 ;
C 99 ; WX 501 ; N Csmall ; B 38 -14 477 448 ;
```

Our font `psbrc8a` features small caps in place of lowercase letters but it has to follow Adobe Standard encoding:

```
C 97 ; WX 556 ; N a ; B 10 0 546 509 ;
C 98 ; WX 556 ; N b ; B 49 0 497 490 ;
C 99 ; WX 556 ; N c ; B 49 -12 512 502 ;
```

This is one of the tricky parts when installing typical `sc & OSF` sets. Fontinst’s encoding vectors expect distinct names for distinct glyphs while the metric files of `sc & OSF` fonts do not provide unique names for optical small caps and hanging figures. The other idiosyncrasy of `sc & OSF` sets is specific to a few font foundries (including Adobe) only: there is no separate upright OSF font so we have to

¹ The correct name of this font is `psbrcj8a`, but we will stick to the naming proposed in `adobe.map` here.

take the upright hanging figures from the small caps font when building virtual fonts.

III.1 The fontinst file

For fontinst, we use the file introduced in the last tutorial as a template and add the features we need. We will create two Latex font families: psb and psbj. The former will provide lining figures while the latter will use the hanging figures of the osf fonts instead. Both families will incorporate optical small caps where available. In the following, all comments concerning the fontinst file will be restricted to those aspects diverging from our template. Please refer to the previous tutorial for a commentary on the original template.

```

1 \input fontinst.sty
2 \substitutesilent{bx}{b}
3 \substitutesilent{sc}{n}
4 \setint{smallcapsscale}{800}
5 \setint{slant}{167}
6 \transformfont{psbr8r}{\reencodefont{8r}{\fromafm{psbr8a}}}
7 \transformfont{psbri8r}{\reencodefont{8r}{\fromafm{psbri8a}}}
8 \transformfont{psbb8r}{\reencodefont{8r}{\fromafm{psbb8a}}}
9 \transformfont{psbbi8r}{\reencodefont{8r}{\fromafm{psbbi8a}}}
10 \transformfont{psbrc8r}{\reencodefont{8r}{\fromafm{psbrc8a}}}
11 \transformfont{psbrij8r}{\reencodefont{8r}{\fromafm{psbrij8a}}}
12 \transformfont{psbbj8r}{\reencodefont{8r}{\fromafm{psbbj8a}}}
13 \transformfont{psbbij8r}{\reencodefont{8r}{\fromafm{psbbij8a}}}

```

The first couple of lines of our template remain unchanged (1–9). After the reencodings inherited from our template, we insert the additional fonts since they need to be reencoded as well (10–13).

```

14 \transformfont{psbro8r}{\slantfont{\int{slant}}\reencodefont{8r}{\fromafm{psbr8a}}}
15 \transformfont{psbbo8r}{\slantfont{\int{slant}}\reencodefont{8r}{\fromafm{psbb8a}}}
16 \transformfont{psbrco8r}{\slantfont{\int{slant}}\reencodefont{8r}{\fromafm{psbrc8a}}}
17 \transformfont{psbboj8r}{\slantfont{\int{slant}}\reencodefont{8r}{\fromafm{psbbj8a}}}

```

In addition to that, we need slanted versions of the new fonts. Slanting the small caps font (16) may seem like a strange thing to do at first since we do not really want to create a slanted small caps shape. But since regular-weight hanging figures are found in the small caps font, we need a slanted version of that as well to provide matching figures for the slanted shape of the psbj family later.

```

18 \installfonts
19 \installfamily{T1}{psb}{}
20 \installfamily{TS1}{psb}{}
21 \installfont{psbr8t}{psbr8r,latin}{t1}{T1}{psb}{m}{n}{}
22 \installfont{psbrc8t}{psbrc8r,unsetnum,kernoff,psbr8r,kernon,latin}{t1}{T1}{psb}{m}{sc}{}

```

The file psbrc8r provides small caps and hanging figures, but we want psb to be a consistent family using lining figures throughout. Therefore, we read psbrc8r first and clear the encoding positions of all figures using commands from a separate metric file, unsetnum.mtx, right after that. This file is listed further down; all it does is clear all figure slots. When adding psbr8r after-

wards, the figure slots of the virtual font `psbrc8t` will be filled using the lining figures found in `psbr8r`. Note that `\installfont` does not overwrite any encoding slots when processing additional metric files, it simply fills vacant slots if it finds suitable glyphs in the next font. This allows us to insert the lining figures of `psbr8r` in the virtual font while the rest of the glyphs including the small caps is taken from `psbrc8r`. As to the encoding vector, we use the regular encoding file `t1.etx` in this case since `psbrc8r` uses standard glyph names for the small caps so that `t1c.etx` would be inappropriate.

There is one more thing we have to take into account: adding a metric file to the `\installfont` command also adds kerning information provided by that file. The problem here is that some of the glyph names in our raw fonts are not unique since the small caps in `psbrc8r` are encoded and labeled just like the lowercase letters in `psbr8r`. The kerning data in `psbr8r`, however, refers to ordinary lowercase letters. Under certain circumstances, misleading kerning data might thus be included in the virtual small caps font `psbrc8t`. To avoid that, we add two auxiliary files provided by `fontinst`, `kernon.mtx` and `kernoff.mtx`, which enable and disable `fontinst`'s `\setkern` command. When added to the input file list as shown above, this will effectively ignore the kerning data in `psbr8r`.

```
23 \installfont{psbro8t}{psbro8r,latin}{t1}{T1}{psb}{m}{s1}{}
24 \installfont{psbri8t}{psbri8r,latin}{t1}{T1}{psb}{m}{it}{}
25 \installfont{psbb8t}{psbb8r,latin}{t1}{T1}{psb}{b}{n}{}
26 \installfont{psbbc8t}{psbb8r,latin}{t1c}{T1}{psb}{b}{sc}{}

```

Optical small caps are available for the regular weight only. For the bold series we have to make do with ‘faked’ small caps, so we use the encoding file `t1c.etx` here (26). The remaining lines for `T1` encoding do not require any adjustments:

```
27 \installfont{psbbo8t}{psbbo8r,latin}{t1}{T1}{psb}{b}{s1}{}
28 \installfont{psbbi8t}{psbbi8r,latin}{t1}{T1}{psb}{b}{it}{}

```

That’s it for `T1` encoding. While `TS1` is primarily intended for symbols complementing `T1`, it includes hanging figures as well. Since the only way to use them is loading the `textcomp` package and typing rather cumbersome text commands like `\textzeroldstyle` it is not very useful to have them in `TS1`. Our `psbj` family will make them the default figures anyway so that they are readily available. But we are being picky. We have put down some hard, cold cash for the `Sabon sc & osf` package and we want to make the most of it. Let’s see how we can put hanging figures in `TS1/psb` as well. As mentioned above, the problem here is that the `osf` fonts use regular glyph names for the hanging figures while `fontinst`'s `TS1` encoding vector references them by `oldstyle` names. Hence we have to turn regular figures – which are in fact hanging figures not encoded as such – into hanging figures. To do that, we need an additional resource provided by `fontinst`, the metric file `resetosf.mtx`. With this in mind, let’s add a section for `TS1` encoding to our `fontinst` file:

```

29 \installfont{psbr8c}{psbr8r,unsetnum,kernoff,psbrc8r,kernon,resetosf,textcomp}{ts1}%
30 {TS1}{psb}{m}{n}{}

```

For the upright fonts, the hanging figures are in fact in the small caps font which complicates the installation even more. But we have dealt with this problem before and the first steps should therefore look familiar: we read `psbr8r`, clear the standard figures using `unsetnum`, and read `psbrc8r`. Since we are dealing with `TS1` here, one additional step is required. We add `resetosf.mtx` to the input file list of this `\installfont` command to rename the figures found in `psbrc8r` (the figures in `psbr8r` have already been discarded by `unsetnum`). `resetosf` will rename the figures to “zerooldstyle” and so on. We also add `kernon.mtx` and `kernoff.mtx` to protect the kerning data. Typing `\textthreeoldstyle` in a Latex file when the `textcomp` package has been loaded would now typeset a proper hanging three.

```

31 \installfont{psbro8c}{psbro8r,unsetnum,kernoff,psbro8r,kernon,resetosf,textcomp}{ts1}%
32 {TS1}{psb}{m}{s1}{}

```

The slanted shape is handled in a similar way because it relies on the figures in the small caps font as well. For the remaining virtual fonts, the installation is simpler. Since the `OSF` fonts already provide hanging figures, all we need to do is rename them for `TS1` encoding by adding `resetosf.mtx`:

```

33 \installfont{psbri8c}{psbrij8r,resetosf,textcomp}{ts1}{TS1}{psb}{m}{it}{}
34 \installfont{psbb8c}{psbbj8r,resetosf,textcomp}{ts1}{TS1}{psb}{b}{n}{}
35 \installfont{psbbo8c}{psbboj8r,resetosf,textcomp}{ts1}{TS1}{psb}{b}{s1}{}
36 \installfont{psbbi8c}{psbbij8r,resetosf,textcomp}{ts1}{TS1}{psb}{b}{it}{}
37 \endinstallfonts

```

This is the first half of our fontinst file which is dealing with the `psb` family. Compared to the template introduced in the previous tutorial it adds optical small caps to `T1` and hanging figures to `TS1` encoding. We will create an additional font family called `psbj` which we want to use hanging figures throughout.

```

38 \installfonts
39 \installfamily{T1}{psbj}{}
40 \installfont{psbrj8t}{psbr8r,unsetnum,kernoff,psbrc8r,kernon,latin}{t1}{T1}{psbj}{m}{n}{}

```

If we want the `psbj` family to incorporate hanging figures, we need to exchange the figure set of the virtual font like we did when creating the regular-weight small caps font above. But this time, we do it the other way around: we read `psbr8r` first, clear the encoding slots of all figures, and add `psbrc8r` afterwards to fill the figure slots using the hanging figures found in `psbrc8r`. Only the figures found in `psbrc8r` will be included in the virtual font as all other encoding slots were already filled by `psbr8r`. Again, care needs to be taken with the kerning data here. The kerning information in `psbrc8r` refers to small caps although the glyphs are encoded as ordinary lowercase letters. Hence we need to add `kernon.mtx` and `kernoff.mtx` to discard the kerning data in `psbrc8r`.

```
41 \installfont{psbrcj8t}{psbrc8r,latin}{t1}{T1}{psbj}{m}{sc{}}
```

The small caps font does not require any modifications this time. `psbrc8r` already contains hanging figures so we can use it as-is. Since `psbrc8r` uses standard glyph names for small caps and hanging figures, we use the regular encoding vector `t1.etx`.

```
42 \installfont{psbroj8t}{psbro8r,unsetnum,kernoff,psbrco8r,kernon,latin}{t1}{T1}{psbj}{m}{sl{}}
```

The slanted shape is straightforward to the upright one: we read `psbro8r`, clear the figures, and add the slanted hanging figures provided by `psbrco8r`. We also toggle `fontinst`'s `\setkern` macro by adding `kernon` and `kernoff`.

```
43 \installfont{psbrij8t}{psbrij8r,latin}{t1}{T1}{psbj}{m}{it{}}
```

Building the italic virtual font is trivial because we have an italic `osF` font with easily accessible hanging figures in the standard slots. Since there are `osF` fonts for all bold shapes as well, they do not require any special modifications either. We simply use the appropriate `osF` fonts instead of the fonts from the basic Sabon package:

```
44 \installfont{psbbj8t}{psbbj8r,latin}{t1}{T1}{psbj}{b}{n{}}
```

```
45 \installfont{psbbcj8t}{psbbj8r,latin}{t1c}{T1}{psbj}{b}{sc{}}
```

We create 'faked' bold small caps using the special `t1c.etx` encoding file because there is no bold small caps font.

```
46 \installfont{psboj8t}{psboj8r,latin}{t1}{T1}{psbj}{b}{sl{}}
```

```
47 \installfont{psbbij8t}{psbbij8r,latin}{t1}{T1}{psbj}{b}{it{}}
```

```
48 \endinstallfonts
```

```
49 \bye
```

This is the complete `fontinst` file for the `NFSS` font families `psb` and `psbj`. It requires the metric file `unsetnum.mtx` which is part of the `fontinst` package. Metric files always begin with `\relax` and enclose all commands in a `metrics` environment. Essentially, `unsetnum.mtx` consists of several `\unsetglyph` commands which clear all figure slots:

```
\relax
\metrics
\unsetglyph{zero}
\unsetglyph{one}
\unsetglyph{two}
\unsetglyph{three}
\unsetglyph{four}
\unsetglyph{five}
\unsetglyph{six}
\unsetglyph{seven}
\unsetglyph{eight}
\unsetglyph{nine}
\endmetrics
```

You probably will have noticed that we did not create `TS1` encoded fonts for the `psbj` family. The reason is quite simple: since `TS1` is not a regular text encoding `TS1/psbj` would be identical to `TS1/psb` anyway. To ensure that the `textcomp`

package works for the psbj family nonetheless, we need to set up some substitutions. Since fontinst does not support family substitutions we cannot create them automatically. We have to write a font definition file manually. The file `ts1psbj.fd` should like this:

```
\ProvidesFile{ts1psbj.fd}
\DeclareFontFamily{TS1}{psbj}{}
\DeclareFontShape{TS1}{psbj}{m}{n}{<-> ssub * psb/m/n} {}
\DeclareFontShape{TS1}{psbj}{m}{sc}{<-> ssub * psb/m/sc} {}
\DeclareFontShape{TS1}{psbj}{m}{sl}{<-> ssub * psb/m/sl} {}
\DeclareFontShape{TS1}{psbj}{m}{it}{<-> ssub * psb/m/it} {}
\DeclareFontShape{TS1}{psbj}{b}{n}{<-> ssub * psb/b/n} {}
\DeclareFontShape{TS1}{psbj}{b}{sc}{<-> ssub * psb/b/sc} {}
\DeclareFontShape{TS1}{psbj}{b}{sl}{<-> ssub * psb/b/sl} {}
\DeclareFontShape{TS1}{psbj}{b}{it}{<-> ssub * psb/b/it} {}
\DeclareFontShape{TS1}{psbj}{bx}{n}{<-> ssub * psb/b/n} {}
\DeclareFontShape{TS1}{psbj}{bx}{sc}{<-> ssub * psb/b/sc} {}
\DeclareFontShape{TS1}{psbj}{bx}{sl}{<-> ssub * psb/b/sl} {}
\DeclareFontShape{TS1}{psbj}{bx}{it}{<-> ssub * psb/b/it} {}
\endinput
```

The syntax of font definition files is explained in the *Latex font selection guide* and will not be discussed in detail here.¹ The main point of this file should be evident: for all series and shapes, we substitute `TS1/psb` for `TS1/psbj` because we did not create virtual fonts for `TS1/psbj`. The `ssub` directive is a silent substitution. For details, see chapter 4 of the *font selection guide*, section 4.4 in particular. With this additional font definition file we now have a fully functional setup for `psb` and `psbj` in `T1` and `TS1` encoding.

III.2 The map file

After running the `fontinst` file through `TeX` and installing the new fonts, we still need to update the map file `psb.map`. We add the following lines for the additional fonts found in the `sc & osf` package:

```
psbrc8r Sabon-RomanSC "TeXBase1Encoding ReEncodeFont" <8r.enc <psbrc8a.pfb
psbrij8r Sabon-ItalicOsF "TeXBase1Encoding ReEncodeFont" <8r.enc <psbrij8a.pfb
psbbj8r Sabon-BoldOsF "TeXBase1Encoding ReEncodeFont" <8r.enc <psbbj8a.pfb
psbbij8r Sabon-BoldItalicOsF "TeXBase1Encoding ReEncodeFont" <8r.enc <psbbij8a.pfb
```

In addition to this, we need slanted versions of the new fonts. For the bold `osf` font this is obvious. Since regular-weight hanging figures are found in the small caps font, we need a slanted version of this font as well to provide matching figures for the slanted shape of the `psbj` family. This leads us to the slanted small caps variant:

```
psbrco8r Sabon-RomanSC "0.167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc <psbrco8a.pfb
psbboj8r Sabon-BoldOsF "0.167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc <psbboj8a.pfb
```

This is the complete map file:

```
psbr8r Sabon-Roman "TeXBase1Encoding ReEncodeFont" <8r.enc <psbr8a.pfb
```

¹ <http://www.ctan.org/tex-archive/macros/latex/doc/fntguide.pdf>

```

psbri8r Sabon-Italic "TeXBase1Encoding ReEncodeFont" <8r.enc <psbri8a.pfb
psbb8r Sabon-Bold "TeXBase1Encoding ReEncodeFont" <8r.enc <psbb8a.pfb
psbbi8r Sabon-BoldItalic "TeXBase1Encoding ReEncodeFont" <8r.enc <psbbi8a.pfb
psbrc8r Sabon-RomanSC "TeXBase1Encoding ReEncodeFont" <8r.enc <psbrc8a.pfb
psbrij8r Sabon-ItalicOsF "TeXBase1Encoding ReEncodeFont" <8r.enc <psbrij8a.pfb
psbbj8r Sabon-BoldOsF "TeXBase1Encoding ReEncodeFont" <8r.enc <psbbj8a.pfb
psbbij8r Sabon-BoldItalicOsF "TeXBase1Encoding ReEncodeFont" <8r.enc <psbbij8a.pfb
psbro8r Sabon-Roman "0.167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc <psbr8a.pfb
psbbo8r Sabon-Bold "0.167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc <psbb8a.pfb
psbrco8r Sabon-RomanSC "0.167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc <psbrc8a.pfb
psbboj8r Sabon-BoldOsF "0.167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc <psbbj8a.pfb

```

III.3 The style file

With two Sabon families at hand, we might want to update `sabon.sty` to make them readily available. We add the two options `oldstyle` and `lining` for the respective font families (6–7) and make hanging figures the default (8). Loading the package with the option `oldstyle` or without any option will set up `psbj` as the default roman family while using the `lining` option will make it select `psb` instead. It might also be handy to have dedicated text commands to switch between the two figure sets. Since such commands will need to work with all font families anyway, let's put them in a stand-alone style file, `nfssexst.sty`, and load that in `sabon.sty` (5):

```

1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{sabon}[2002/05/12 v1.1 Adobe Sabon]
3 \RequirePackage[T1]{fontenc}
4 \RequirePackage{textcomp}
5 \RequirePackage{nfssexst}
6 \DeclareOption{oldstyle}{\renewcommand*\rmdefault}{psbj}}
7 \DeclareOption{lining}{\renewcommand*\rmdefault}{psb}}
8 \ExecuteOptions{oldstyle}
9 \ProcessOptions
10 \endinput

```

The style file `nfssexst.sty` might look like this:

```

1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{nfssexst}[2003/03/14 v1.2 Experimental NFSS Extensions]
3 \newcommand*\exfs@tempa{}
4 \newcommand*\exfs@tempb{}
5 \newcommand*\exfs@try@family[1]{%
6   \let\exfs@tempa\relax
7   \begingroup
8     \fontfamily{#1}\try@load@fontshape
9     \expandafter\ifx\csname\curr@fontshape\endcsname\relax
10    \PackageWarning{nfssexst}{%
11      Font family '\f@encoding/#1' not available\MessageBreak
12      Ignoring font switch}%
13   \else
14     \gdef\exfs@tempa{\fontfamily{#1}\selectfont}%
15   \fi
16 \endgroup
17 \exfs@tempa}

```

This is an outline for a command that makes use of a few NFSS internals to switch to a specific family if and only if it is available. Essentially, we try to load the requested family in the current encoding (8). If this succeeds, we set up a macro (14) to be expanded later that will actually switch font families; if not, we print a warning message (10–12) and do nothing.

```

18 \def\exfs@get@base#1#2#3#4\@nil{#1#2#3}%
19 \DeclareRobustCommand{\lnstyle}{%
20   \not@math@alphabet\lnstyle\relax
21   \exfs@try@family{\expandafter\exfs@get@base\family\@nil}}
22 \DeclareRobustCommand{\osstyle}{%
23   \not@math@alphabet\osstyle\relax
24   \exfs@try@family{\expandafter\exfs@get@base\family\@nil j}}

```

The macros `\lnstyle` and `\osstyle` switch to lining and hanging (‘old style’) figures respectively. They are like `\bfseries` or `\itshape`. Internally, they will take the first three letters of the current NFSS font family name (18), append a letter to it where appropriate (none for lining figures, `j` for hanging figures), and call `\exfs@try@family`. Even though this mechanism is rather simple-minded, it should work just fine for all fonts set up properly according to the Fontname scheme.

```

25 \DeclareTextFontCommand{\textln}{\lnstyle}
26 \DeclareTextFontCommand{\textos}{\osstyle}
27 \endinput

```

The corresponding text commands, `\textln` and `\textos`, take one mandatory argument and can be employed like `\textbf` or `\textit`.

III.4 Fonts supplied with Tex

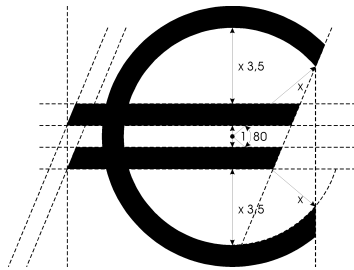
The standard Postscript fonts supplied with most Tex distributions do not include optical small caps, nor do they include hanging figures. The default typeface of both plain Tex and Latex however, Computer Modern Roman, does include such glyphs. Unfortunately, the design of the small caps is flawed. Their height corresponds to what you usually end up with when creating mechanical small caps. Being too tall, these small caps hardly blend in with lowercase text at all, even though their color matches that of the lowercase alphabet.

Hanging figures are included in Computer Modern as well, but they are hidden in some of the math fonts. The only way to use them with the default setup is rather cumbersome: the command `\oldstylenums{}` will take the numbers to be typeset as hanging figures as an argument. There is a set of virtual fonts for the European Computer Modern fonts which make these hanging figures the default in Tex’s text mode so that they are readily available. These fonts are provided in the `ECO` package available from CTAN.¹ Please refer to the package documentation for installation and usage instructions. Since this package essentially consists of a set of virtual fonts, it should also work in conjunction with the `CM-super` fonts mentioned in section I.6.

¹ <http://www.ctan.org/tex-archive/fonts/eco/>

TUTORIAL IV

THE EURO CURRENCY SYMBOL



While the euro symbol has been supported by LaTeX for quite some time – it is included in `rs1` encoding and the `textcomp` package provides the corresponding text command `\texteuro` – the real problem is getting fonts that provide this glyph and setting them up accordingly. You might want to read this tutorial even if you are not affected by this particular issue, because it

deals with some generic encoding problems that you may encounter in a different context as well. There is a bit more to updating a font than drawing a euro symbol and putting it in the font. It has to be properly encoded as well. Since the euro symbol is not defined in Adobe Standard encoding, it can normally only be included as an uncoded glyph in regular Postscript text fonts. An uncoded glyph is only accessible after reencoding and assigning it to a valid encoding position. Some font foundries decided to follow this path in order to conform to Adobe Standard encoding. Others preferred to drop some supposedly rarely used glyph and put the euro symbol in its encoding position instead. While this violates the encoding standard, it can be more convenient under certain circumstances. In the following, we will explore ways to handle both situations cleanly. Finally, we will learn how to take the euro symbol from an external font if none is provided by the text font itself.

iv.1 Uncoded euro symbol

While Adobe used to be rather inattentive to the problem at first, the foundry is finally updating their typeface portfolio by gradually adding matching euro symbols to their fonts – a process that has been promoted by the introduction of the OpenType font format. Recent releases of Adobe Garamond, for example, already ship with matching euro symbols. A quick look at the `afm` file shows that in this case, the foundry decided to handle the encoding problem in a strict manner. The new symbol is correctly labeled as “Euro” but it is not encoded by default as that would violate Adobe Standard encoding. An encoding slot number of -1 tells us that the glyph was not assigned to any encoding position:

```
C -1 ; WX 572 ; N Euro ; B -13 -14 542 640 ;
```

In order to access it, we need to reencode the font and assign the glyph “Euro” to a valid encoding position. The standard procedure we have been pursuing in this guide involves reencoding all fonts to Tex Base 1 encoding anyway precisely

because of cases like this one. By reencoding all base fonts to Tex Base 1 encoding, we ensure that all glyphs our virtual fonts rely on are properly encoded in the raw fonts we use as their basis. But we have to keep in mind that older versions of Tex Base 1 encoding did not include the euro symbol either. The previous release of fontinst, version 1.8, came with an encoding vector that is not suitable for our situation for this very reason. You can verify that by running the file `8r.etx` through Latex to create a documented listing of the encoding vector as follows:

```
latex 8r.etx
```

Now take a look at the DVI file `8r.dvi`: if the version number of this file is 1.801 (dated June 29, 1998), it does not include the euro symbol. The best way to solve this problem is updating fontinst to the latest release which ships with an updated encoding vector.¹ After that, you can install the fonts as usual. Note, however, that you will need a matching version of `8r.enc` as well, so that dvips and pdftex can use the symbol. This file is distributed separately and not included in the fontinst release.² In the following, we will create our own updated versions of `8r.etx` and `8r.enc`. This is merely intended as an illustrating of how to deal with a typical encoding problem. If you simply want to get access to an uncoded euro glyph, upgrade to fontinst 1.9, update `8r.enc`, and install the fonts as usual. You might want to skip the next paragraphs and continue reading with section IV.2 on page 46 in this case. If you want to learn more about fontinst’s encoding vectors, read on.

First, we create a copy of the file `8r.etx` as provided by fontinst 1.8. The updated encoding vector of the new fontinst release puts the euro symbol in slot 128. We will do the same to ensure that our vector remains compatible with the official distribution. Let’s take a look at the relevant part of `8r.etx`:

```
624 \setslot{asciitilde}
625   \comment{The ASCII tilde '\textasciitilde'.
626     This is included for compatibility with typewriter fonts used
627     for computer listings.}
628 \endsetslot
629
630 \comment{The following 32 slots, 128--159, are based on Windows ANSI.}
631
632 \nextslot{130}
633 \setslot{quotesinglbase}
634   \comment{A German single quote mark '\quotesinglbase' similar to a comma,
635     but with different sidebearings.}
636 \endsetslot
```

Slot 126 defines “asciitilde”, slots 127–129 are empty, and slot 130 defines the lower single quotation mark “quotesinglbase”. The slot number is automatically incremented by one for each `\setslot` command, but if some slots are left

¹ <http://www.ctan.org/tex-archive/fonts/utilities/fontinst/>

² <http://www.ctan.org/tex-archive/info/fontname/8r.enc>

empty the slot has to be set explicitly with a `\nextslot` command. This is done for “quotesinglbase” above. We want to add the euro symbol in slot 128, so we add the following:

```
630 \comment{The following 32 slots, 128--159, are based on Windows ANSI.}
    \nextslot{128}
    \setslot{Euro}
        \comment{The euro currency symbol '\texteuro' .}
    \endsetslot
632 \nextslot{130}
633 \setslot{quotesinglbase}
634     \comment{A German single quote mark '\quotesinglbase' similar to a comma,
635         but with different sidebearings.}
636 \endsetslot
```

Since slot 127 is empty and the last slot defined was 126 we need to set the slot explicitly by adding `\nextslot` before actually defining the encoding position. When defining the slot, keep in mind that the glyph names are case sensitive; “euro” is not equivalent to “Euro”. We also add an explanation so that the commented listing of the encoding vector provides a meaningful explanation. This is all we need. It might be a good idea to update `\title` and `\date` at the beginning of the file to avoid any confusion. Finally, we install this file in the branch `tex/fontinst/base/` of the local Tex tree. If our system has been set up as recommended in the first tutorial, fontinst will now pick up our updated encoding vector. Now we need a version of `8r.enc` that matches our `8r.etx`. This is what the relevant part of `8r.enc` looks like:

```
71 % 0x70
72 /p /q /r /s /t /u /v /w
73 /x /y /z /braceleft /bar /braceright /asciitilde
74 /.notdef
75 % 0x80
76 /.notdef /.notdef /quotesinglbase /florin
77 /quotedblbase /ellipsis /dagger /daggerdbl
78 /circumflex /perthousand /Scaron /guilsinglleft
79 /OE /.notdef /.notdef /.notdef
```

Note that in Postscript encoding vectors empty slots are marked “`.notdef`”. We can spot the same pattern: “`asciitilde`” in slot 126 is followed by three empty slots (127–129) and finally “`quotesinglbase`” in slot 130. We count the slots and add “Euro” in slot 128 (indicated in hexadecimal notation as ‘0x80’ here):

```
71 % 0x70
72 /p /q /r /s /t /u /v /w
73 /x /y /z /braceleft /bar /braceright /asciitilde
74 /.notdef
75 % 0x80
76 /Euro /.notdef /quotesinglbase /florin
77 /quotedblbase /ellipsis /dagger /daggerdbl
78 /circumflex /perthousand /Scaron /guilsinglleft
79 /OE /.notdef /.notdef /.notdef
```

After that, we move our modified `8r.enc` to `dvips/base/` in the local TeX tree and update the `kpathsea` file databases by running `texhash`. Our system is now ready for the euro. Since reencoding all text fonts to TeX Base 1 encoding is part of our regular installation routine anyway, the `fontinst` file does not need any adjustments. The reencoding is performed as usual:

```
\transformfont{padr8r}{\reencodefont{8r}{\fromafm{padr8a}}}
\transformfont{padi8r}{\reencodefont{8r}{\fromafm{padi8a}}}
\transformfont{padb8r}{\reencodefont{8r}{\fromafm{padb8a}}}
\transformfont{padbi8r}{\reencodefont{8r}{\fromafm{padbi8a}}}
```

The new `8r` encoding vector will ensure that the euro symbol is available in all TeX Base 1 encoded raw fonts, so we can simply use them to build `TS1` encoded virtual fonts:

```
\installfont{padr8c}{padr8r,textcomp}{ts1}{TS1}{pad}{m}{n}{}
\installfont{padi8c}{padi8r,textcomp}{ts1}{TS1}{pad}{m}{it}{}
\installfont{padb8c}{padb8r,textcomp}{ts1}{TS1}{pad}{b}{n}{}
\installfont{padbi8c}{padbi8r,textcomp}{ts1}{TS1}{pad}{b}{it}{}

```

After installing the fonts and creating a map file as usual, the euro symbol will be available as `\texteuro` when loading the `textcomp` package.

iv.2 Euro symbol encoded as currency symbol

Bitstream was one of the first type foundries to update their font collection and add a matching euro symbol to all fonts. When looking at the fonts, the first thing we notice is that the foundry decided to encode the euro symbol as the generic currency symbol € . The reasoning behind this is that you can access the symbol without reencoding the font. Since the generic currency symbol is hardly ever used anyway, it is no surprise that this particular glyph was dropped. We could install Bitstream fonts as usual and use `\textcurrency` instead of `\texteuro` to access the euro symbol, but that would imply keeping the idiosyncrasies of a given font in mind while writing, and modifying the text when changing the typeface – not quite what one would expect when working with a high-level markup language like LaTeX. When taking a closer look at the `pfb` and `afm` files, we can see that the fonts in fact contain two euro symbols. One of them is uncoded (slot -1) and labeled as “Euro”:

```
C -1 ; WX 556 ; N Euro ; B 6 -12 513 697 ;
```

The other one is found in encoding slot 168, that is, it is encoded as the currency symbol and named accordingly. To verify that, we have to take a look at the `pfb` files in a font viewer or a font editor. Since the euro symbol is both encoded and labeled just like a currency symbol, there is no way to tell the difference by looking at the `afm` file only:

§ ₁₆₇	€ ₁₆₈	' ₁₆₉
• ₁₆₃	, ₁₈₄	” ₁₈₅
. ₁₉₉	.. ₂₀₀	

```
C 168 ; WX 556 ; N currency ; B 6 -12 513 697 ;
```

If we want a readily available euro symbol (and one that is available *as such*), we have two options in this case. Either we reencode the font and assign the uncoded euro symbol to a valid encoding position or we use the already encoded euro symbol found in the slot of the currency symbol and move it to the proper encoding position. The former case was already discussed above, let's now investigate the latter.

The best way to move the glyph to a different slot is resetting it when creating the TS1 encoded virtual font. We use an approach that is functionally equivalent to the way we have reset the hanging figures in the previous tutorial. The appropriate low-level commands that set the glyph go in a dedicated metric file, `reseteur.mtx`, which we have to create ourselves:

```

1 \relax
2 \metrics
3 \resetglyph{euro}
4   \glyph{currency}{1000}
5 \endsetglyph
6 \setleftkerning{euro}{currency}{1000}
7 \unsetglyph{currency}
8 \endmetrics

```

We reset the glyph “euro” based on the glyph “currency” scaled to its full size (3–5), adjust the kerning on either side of “euro” to match that of “currency” (6) and finally unset the glyph “currency” (7) because there is no such thing as a currency symbol in this font. In the fontinst file, we include the metric file `reseteur.mtx` in the file list of the respective `\installfont` command right after the metrics for this font have been read. This might look as follows:

```
\installfont{bsbr8c}{bsbr8r,reseteur,textcomp}{ts1}{TS1}{bsb}{m}{n}{}
```

We only need to do this for the TS1 encoded virtual fonts as T1 does not include the euro symbol. Apart from that, the fontinst file does not need any adjustments.

iv.3 Euro symbol taken from external symbol font

Let's go back to our install of Sabon to see if we can get euro support for Sabon as well. The font itself does not include a euro symbol at all so all we can do is take it from an external font. While some other font foundries at least provide special symbol fonts containing a collection of matching euro glyphs for all typefaces that have not been updated yet, Adobe merely offers a set of generic euro fonts containing glyphs that do not really match any typeface at all.¹ From a typographical perspective, this is a desperate workaround. However, lacking a matching euro symbol, we do not have a choice. The Adobe Euro fonts come in three flavors: serif (Euro Serif), sans serif (Euro Sans), and condensed sans serif (Euro Mono, intended for use with monospaced fonts). Each family consists of regular, regular italic, bold, and bold italic fonts. Instead of using a serif euro

¹ <http://www.adobe.com/type/eurofont.html>

that does not match our typeface we will use the sans serif design which has a more generic look that adheres to the shape of the reference design of the European Commission. Granted, this one does not match our typeface either – but at least it does not pretend to do so.

Now that we are aware of the most common encoding pitfalls, we inspect the afm files first before proceeding with the installation. The Euro fonts put the euro symbol in all encoding positions. When looking at the afm file, we can see that the fonts use a font specific encoding and that the glyphs are labeled as “Euro” with a consecutive number appended to the name:

```
C 33 ; WX 750 ; N Euro.001 ; B 10 -12 709 685 ;
C 34 ; WX 750 ; N Euro.002 ; B 10 -12 709 685 ;
C 35 ; WX 750 ; N Euro.003 ; B 10 -12 709 685 ;
C 36 ; WX 750 ; N Euro.004 ; B 10 -12 709 685 ;
C 37 ; WX 750 ; N Euro.005 ; B 10 -12 709 685 ;
C 38 ; WX 750 ; N Euro.006 ; B 10 -12 709 685 ;
C 39 ; WX 750 ; N Euro.007 ; B 10 -12 709 685 ;
C 40 ; WX 750 ; N Euro.008 ; B 10 -12 709 685 ;
C 41 ; WX 750 ; N Euro.009 ; B 10 -12 709 685 ;
```

On further inspection, we find two additional glyphs. There is a glyph labeled as “Euro” in slot 128 as well as an uncoded glyph labeled “uni20AC”:

```
C 128 ; WX 750 ; N Euro ; B 10 -12 709 685 ;
C -1 ; WX 750 ; N uni20AC ; B 10 -12 709 685 ;
```

The number 20AC is 8364 in hexadecimal. This is the encoding position of the euro symbol in Unicode encoding, hence the string “uni20AC”. If nothing else, one thing is for sure: someone was trying to make sure that every application out there would be able to access that euro symbol. Fortunately, this covers our situation as well. We need a glyph that is both properly encoded and labeled as “Euro”; the encoding position does not matter since we will include it in a virtual font using a different encoding anyway. The one in slot 128 fits our needs perfectly. In practice, this means that we can simply add the file name to the input file list of an `\installfont` command when creating `TS1` encoded virtual fonts with `fontinst`. This time no reencoding or renaming is required. The relevant section of our `fontinst` file for Sabon would look as follows:

```
\installfont{psbr8c}{psbr8r,zpeurs,textcomp}{ts1}{TS1}{psb}{m}{n}{}
\installfont{psbro8c}{psbro8r,zpeuros,textcomp}{ts1}{TS1}{psb}{m}{s1}{}
\installfont{psbri8c}{psbri8r,zpeuris,textcomp}{ts1}{TS1}{psb}{m}{it}{}
\installfont{psbb8c}{psbb8r,zpeubs,textcomp}{ts1}{TS1}{psb}{b}{n}{}
\installfont{psbbo8c}{psbbo8r,zpeubos,textcomp}{ts1}{TS1}{psb}{b}{s1}{}
\installfont{psbbi8c}{psbbi8r,zpeubis,textcomp}{ts1}{TS1}{psb}{b}{it}{}

```

Since the Adobe Euro fonts are non-standard, their naming is non-standard as well. We will discuss that in more detail below. Before running this file, we need to copy the properly named afm files of the Adobe Euro fonts to the working directory so that `fontinst` will find them. For the euro glyph to be available later, the Euro fonts need to be installed in the usual way so that `TeX` as well as `pdftex`, `dvips`, and `xdvi` are able to use them. Because the above `fontinst` file depends

on this installation, it makes sense to do it first. Since the installation of symbol fonts differs from that of regular text fonts, we will take a look at the required steps. The Euro font package¹ will provide us with the following set of files:

```
_1____.afm  _1i____.afm  _1b____.afm  _1bi____.afm
_1____.inf  _1i____.inf  _1b____.inf  _1bi____.inf
_1____.pfb  _1i____.pfb  _1b____.pfb  _1bi____.pfb
_1____.pfm  _1i____.pfm  _1b____.pfm  _1bi____.pfm

_2____.afm  _2i____.afm  _2b____.afm  _2bi____.afm
_2____.inf  _2i____.inf  _2b____.inf  _2bi____.inf
_2____.pfb  _2i____.pfb  _2b____.pfb  _2bi____.pfb
_2____.pfm  _2i____.pfm  _2b____.pfm  _2bi____.pfm

_3____.afm  _3i____.afm  _3b____.afm  _3bi____.afm
_3____.inf  _3i____.inf  _3b____.inf  _3bi____.inf
_3____.pfb  _3i____.pfb  _3b____.pfb  _3bi____.pfb
_3____.pfm  _3i____.pfm  _3b____.pfm  _3bi____.pfm
```

The Fontname map file `adobe.map` defines the following names for these fonts:

<code>zpeur</code>	EuroSerif-Regular	A	916	<u> 3 </u>
<code>zpeub</code>	EuroSerif-Bold	A	916	<u> 3b </u>
<code>zpeubi</code>	EuroSerif-BoldItalic	A	916	<u> 3bi </u>
<code>zpeuri</code>	EuroSerif-Italic	A	916	<u> 3i </u>
<code>zpeurs</code>	EuroSans-Regular	A	916	<u> 1 </u>
<code>zpeubs</code>	EuroSans-Bold	A	916	<u> 1b </u>
<code>zpeubis</code>	EuroSans-BoldItalic	A	916	<u> 1bi </u>
<code>zpeuris</code>	EuroSans-Italic	A	916	<u> 1i </u>
<code>zpeurt</code>	EuroMono-Regular	A	916	<u> 2 </u>
<code>zpeubt</code>	EuroMono-Bold	A	916	<u> 2b </u>
<code>zpeubit</code>	EuroMono-BoldItalic	A	916	<u> 2bi </u>
<code>zpeurit</code>	EuroMono-Italic	A	916	<u> 2i </u>

We select all `afm` and all `pfb` files, rename them, and start off with the following file set:

```
zpeur.afm      zpeuri.afm      zpeub.afm      zpeubi.afm
zpeur.pfb      zpeuri.pfb      zpeub.pfb      zpeubi.pfb
zpeurs.afm     zpeuris.afm     zpeubs.afm     zpeubis.afm
zpeurs.pfb     zpeuris.pfb     zpeubs.pfb     zpeubis.pfb
zpeurt.afm     zpeurit.afm     zpeubt.afm     zpeubit.afm
zpeurt.pfb     zpeurit.pfb     zpeubt.pfb     zpeubit.pfb
```

As we do not really need `fontinst` when dealing with symbol fonts, we simply run `afm2tfm` on each `afm` file to create a corresponding `tfm` file for `TeX`:

```
afm2tfm zpeur.afm  zpeur.tfm
afm2tfm zpeuri.afm zpeuri.tfm
afm2tfm zpeub.afm  zpeub.tfm
afm2tfm zpeubi.afm zpeubi.tfm
afm2tfm zpeurs.afm zpeurs.tfm
afm2tfm zpeuris.afm zpeuris.tfm
afm2tfm zpeubs.afm zpeubs.tfm
afm2tfm zpeubis.afm zpeubis.tfm
afm2tfm zpeurt.afm zpeurt.tfm
afm2tfm zpeurit.afm zpeurit.tfm
```

¹ <http://www.adobe.com/type/eurofont.html>

```
afm2tfm zpeubt.afm zpeubt.tfm
afm2tfm zpeubit.afm zpeubit.tfm
```

We also need slanted versions of all upright fonts. As slant factor, we use the generic value 0.167:

```
afm2tfm zpeur.afm -s 0.167 zpeuro.tfm
afm2tfm zpeub.afm -s 0.167 zpeubo.tfm
afm2tfm zpeurs.afm -s 0.167 zpeuros.tfm
afm2tfm zpeubs.afm -s 0.167 zpeubos.tfm
afm2tfm zpeurt.afm -s 0.167 zpeurot.tfm
afm2tfm zpeubt.afm -s 0.167 zpeubot.tfm
```

In addition to that, we need a map file for dvips. Map files for symbol fonts are simpler than those for text fonts because the fonts are not reencoded. Therefore, there will be no “ReEncodeFont” instruction and no encoding vector. The first lines of `peu.map` look like this:

```
zpeur EuroSerif-Regular <zpeur.pfb
zpeuri EuroSerif-Italic <zpeuri.pfb
zpeub EuroSerif-Bold <zpeub.pfb
zpeubi EuroSerif-BoldItalic <zpeubi.pfb
zpeurs EuroSans-Regular <zpeurs.pfb
zpeuris EuroSans-Italic <zpeuris.pfb
zpeubs EuroSans-Bold <zpeubs.pfb
zpeubis EuroSans-BoldItalic <zpeubis.pfb
zpeurt EuroMono-Regular <zpeurt.pfb
zpeurit EuroMono-Italic <zpeurit.pfb
zpeubt EuroMono-Bold <zpeubt.pfb
zpeubit EuroMono-BoldItalic <zpeubit.pfb
```

We also need to add “SlantFont” instructions for all slanted shapes:

```
zpeuro EuroSerif-Regular "0.167 SlantFont" <zpeur.pfb
zpeubo EuroSerif-Bold "0.167 SlantFont" <zpeub.pfb
zpeuros EuroSans-Regular "0.167 SlantFont" <zpeurs.pfb
zpeubos EuroSans-Bold "0.167 SlantFont" <zpeubs.pfb
zpeurot EuroMono-Regular "0.167 SlantFont" <zpeurt.pfb
zpeubot EuroMono-Bold "0.167 SlantFont" <zpeubt.pfb
```

While we are at it, let’s also write some font definition files for Latex. These are not required if the fonts are only referenced by other virtual fonts, but they will allow us the access the Euro fonts directly in any Latex file. The syntax of the commands used in font definition files is explained in the Latex font selection guide mentioned in the introduction. Our font definition file for Euro Serif, `upeu.fd`, should look like this:

```
\ProvidesFile{upeu.fd}
\DeclareFontFamily{U}{peu}{}
\DeclareFontShape{U}{peu}{m}{n}{<-> zpeur} {}
\DeclareFontShape{U}{peu}{m}{sc}{<-> ssub * peu/m/n} {}
\DeclareFontShape{U}{peu}{m}{sl}{<-> zpeuro} {}
\DeclareFontShape{U}{peu}{m}{it}{<-> zpeuri} {}
\DeclareFontShape{U}{peu}{b}{n}{<-> zpeub} {}
\DeclareFontShape{U}{peu}{b}{sc}{<-> ssub * peu/b/n} {}
\DeclareFontShape{U}{peu}{b}{sl}{<-> zpeubo} {}
\DeclareFontShape{U}{peu}{b}{it}{<-> zpeubi} {}
```



```

\DeclareFontShape{U}{peu}{bx}{n}{<->ssub*peu/b/n}{}
\DeclareFontShape{U}{peu}{bx}{sc}{<->ssub*peu/b/n}{}
\DeclareFontShape{U}{peu}{bx}{sl}{<->ssub*peu/b/sl}{}
\DeclareFontShape{U}{peu}{bx}{it}{<->ssub*peu/b/it}{}
\endinput

```

For Euro Sans, `upeus.fd`:

```

\ProvidesFile{upeus.fd}
\DeclareFontFamily{U}{peus}{}
\DeclareFontShape{U}{peus}{m}{n}{<->zpeurs}{}
\DeclareFontShape{U}{peus}{m}{sc}{<->ssub*peus/m/n}{}
\DeclareFontShape{U}{peus}{m}{sl}{<->zpeuros}{}
\DeclareFontShape{U}{peus}{m}{it}{<->zpeuris}{}
\DeclareFontShape{U}{peus}{b}{n}{<->zpeubs}{}
\DeclareFontShape{U}{peus}{b}{sc}{<->ssub*peus/b/n}{}
\DeclareFontShape{U}{peus}{b}{sl}{<->zpeubos}{}
\DeclareFontShape{U}{peus}{b}{it}{<->zpeubis}{}
\DeclareFontShape{U}{peus}{bx}{n}{<->ssub*peus/b/n}{}
\DeclareFontShape{U}{peus}{bx}{sc}{<->ssub*peus/b/n}{}
\DeclareFontShape{U}{peus}{bx}{sl}{<->ssub*peus/b/sl}{}
\DeclareFontShape{U}{peus}{bx}{it}{<->ssub*peus/b/it}{}
\endinput

```

And for Euro Mono, `upeut.fd`:

```

\ProvidesFile{upeut.fd}
\DeclareFontFamily{U}{peut}{}
\DeclareFontShape{U}{peut}{m}{n}{<->zpeurt}{}
\DeclareFontShape{U}{peut}{m}{sc}{<->ssub*peut/m/n}{}
\DeclareFontShape{U}{peut}{m}{sl}{<->zpeurot}{}
\DeclareFontShape{U}{peut}{m}{it}{<->zpeurit}{}
\DeclareFontShape{U}{peut}{b}{n}{<->zpeubt}{}
\DeclareFontShape{U}{peut}{b}{sc}{<->ssub*peut/b/n}{}
\DeclareFontShape{U}{peut}{b}{sl}{<->zpeubot}{}
\DeclareFontShape{U}{peut}{b}{it}{<->zpeubit}{}
\DeclareFontShape{U}{peut}{bx}{n}{<->ssub*peut/b/n}{}
\DeclareFontShape{U}{peut}{bx}{sc}{<->ssub*peut/b/n}{}
\DeclareFontShape{U}{peut}{bx}{sl}{<->ssub*peut/b/sl}{}
\DeclareFontShape{U}{peut}{bx}{it}{<->ssub*peut/b/it}{}
\endinput

```

We install the map file `peu.map` as well as all `afm`, `tfm`, `pfb`, and `fd` files in the local TeX tree as explained in the first tutorial and add `peu.map` to the configuration files for `pdftex`, `dvips`, and `xdvi`. Finally, we run `texhash`. The euro symbol can now be used in virtual fonts. Since we have font definition files for LaTeX as well, we could also access it in any LaTeX file with a construct like this one:

```
{\fontencoding{U}\fontfamily{peu}\selectfont\char 128}
```

So let's make that a generic euro package, `peufonts.sty`, for use with all fonts that do not provide a native euro symbol:

```

1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{peufonts}[2002/10/25 v1.0 Adobe Euro Fonts]
3 \RequirePackage{textcomp}
4 \DeclareRobustCommand{\eurm}{\%

```

```

5 \fontencoding{U}\fontfamily{peu}\selectfont\char 128}
6 \DeclareRobustCommand{\eursf}{%
7 \fontencoding{U}\fontfamily{peus}\selectfont\char 128}}
8 \DeclareRobustCommand{\eurtt}{%
9 \fontencoding{U}\fontfamily{peut}\selectfont\char 128}}

```

We define three commands, `\eurrm`, `\eursf`, and `\eurtt`, which typeset a serif, sans serif, and monospaced euro symbol respectively. Note the additional braces to keep the font change local.

```

10 \DeclareOption{sans}{\def\eur\eursf}
11 \DeclareOption{serif}{\def\eur\eurrm}
12 \DeclareOption{mono}{\def\eur\eurtt}
13 \DeclareOption{textcomp}{%
14 \PackageInfo{peufonts}{Hijacking '\string\texteuro'}%
15 \def\texteuro{\eur}}
16 \ExecuteOptions{sans}
17 \ProcessOptions
18 \endinput

```

We also provide `\eur` along with three options controlling whether it uses the serif, sans serif, or monospaced euro symbol; `sans` is set up as the default in line 19. The option `textcomp` will hijack the text command `\texteuro` as provided by the `textcomp` package. This is very handy when using the `inputenc` package with Latin 9 (ISO8859-15) as input encoding and entering the euro symbol directly, as `inputenc` uses `\texteuro` internally. With this option, we may also type `\texteuro` or simply € in the input file to typeset a euro symbol. For this to work, `inputenc` has to be loaded before this package. Please keep in mind that this is a global redefinition affecting all text fonts. We do not activate it by default as some fonts may provide a native euro symbol. We also write a message to the log when redefining `\texteuro` and request `textcomp` in line 3 so that it is loaded before `peufonts`.

iv.4 Euro symbol taken from external text font

There is yet another way to get the euro symbol for a font that does not provide one by default. Suppose we have an external text font including a euro symbol that would go reasonably well with our copy of Sabon. If this euro symbol is uncoded but labeled correctly, we could simply add the text font to the input file list of the respective `\installfont` commands as shown in section iv.3 and then proceed as outlined in section iv.1. What if it is encoded as the currency symbol in the external text font? In this case, we take an approach that is based on section iv.2 with some minor adjustments. Let's assume we have a copy of Bitstream Classical Garamond. Since Classical Garamond is Bitstream's take on Sabon, the euro symbol of this typeface will obviously go quite well with our install of Sabon. The syntax of the `\installfont` commands will look like this:

```
\installfont{psbr8c}{psbr8r,unsetcur,bsbr8r,reseteur,psbr8r,textcomp}{ts1}{TS1}{bsb}{m}{n}{}
```

psb is Adobe Sabon, bsb is Bitstream Classical Garamond, and `reseteur.mtx` has been discussed in section IV.2. In this case, we need an additional metric file, called `unsetcur.mtx` here, that clears the currency slot before `bsbr8r.afm` is read. Without this additional step, the euro symbol found in the currency slot of `bsbr8r.afm` would be discarded as `psbr8r.afm` has already provided this symbol. `reseteur.mtx` would then move the currency symbol found in `psbr8r.afm` to the euro slot, which is obviously not what we want. We need to clear the currency slot using `unsetcur.mtx`, which is quite simple:

```
\relax
\metrics
\unsetglyph{currency}
\endmetrics
```

With this additional resource, what happens is this: `psbr8r.afm` is read and processed, the currency slot is cleared by `unsetcur.mtx`, then `bsbr8r.afm` is read, filling the currency slot with its euro glyph (which is encoded as the currency symbol in `bsbr8r.afm`). Our metric file `reseteur.mtx` will then move the euro symbol found in `bsbr8r.afm` to the euro slot and clear the currency slot. After that, we read `psbr8r.afm` again to get the original Adobe Sabon currency symbol of back. Our virtual font will now contain all glyphs found in Adobe Sabon plus the euro symbol of Bitstream Classical Garamond, all properly encoded. Note that, for this to work, we need a complete install of Bitstream Classical Garamond, including map files for `dvips` and `pdftex`, in addition to the steps outlined above.

TUTORIAL V

EXPERT FONT SETS, REGULAR SETUP

Expert fonts are complements to be used in conjunction with regular text fonts. They usually contain optical small caps, additional sets of figures – hanging, inferior, superior –, the f-ligatures ff, fi, fl, ffi, and ffl, plus a few text fractions and some other symbols. Since they are companion fonts only, which do not contain the regular uppercase and lowercase alphabet, they are not useful on their own. To employ them in a sensible way we need the basic text fonts as well. In this tutorial, we will install the complete Monotype Janson font set as provided by the base and the expert package offered by Agfa Monotype. The base package contains four text fonts (regular, regular italic, bold, bold italic):

```
jan____.afm   jani____.afm   janb____.afm   janbi____.afm
jan____.inf   jani____.inf   janb____.inf   janbi____.inf
jan____.pfb   jani____.pfb   janb____.pfb   janbi____.pfb
jan____.pfm   jani____.pfm   janb____.pfm   janbi____.pfm
```

The expert package adds the corresponding expert fonts:

```
jny____.afm   jnyi____.afm   jnyb____.afm   jnybi____.afm
jny____.inf   jnyi____.inf   jnyb____.inf   jnybi____.inf
jny____.pfb   jnyi____.pfb   jnyb____.pfb   jnybi____.pfb
jny____.pfm   jnyi____.pfm   jnyb____.pfm   jnybi____.pfm
```

When talking about “expert font sets” in this tutorial, we are referring to all of the above (base plus expert package). The proper file names for Monotype Janson are given in `monotype.map`. Expert fonts have essentially the same file name as the corresponding text fonts, but their encoding code is 8x instead of 8a for Adobe Standard encoding. After renaming the files, we start off with the following file set:

```
mjnr8a.afm   mjnr8a.afm   mjnr8a.afm   mjnr8a.afm
mjnr8a.pfb   mjnr8a.pfb   mjnr8a.pfb   mjnr8a.pfb

mjnr8x.afm   mjnr8x.afm   mjnr8x.afm   mjnr8x.afm
mjnr8x.pfb   mjnr8x.pfb   mjnr8x.pfb   mjnr8x.pfb
```

There are two ways to install an expert font set. Apart from writing a verbose fontinst file using low-level commands we may also use the `\latinfamily` macro. We will take a look at the latter case first and proceed with a verbose fontinst file afterwards.

v.1 Basic fontinst file

As usual, our file begins with a typical header setting up some common font substitutions (2–3). While the Janson expert package provides optical small caps for the regular weight, the bold expert fonts do not contain optical small

caps. For the bold series, we have to make do with mechanical small caps. The `\latinfamily` macro will take care of that automatically. All we need to do is define a scale factor of 0.72 (4):

```
1 \input fontinst.sty
2 \substitutesilent{bx}{b}
3 \substitutesilent{sc}{n}
4 \setint{smallcapsscale}{720}
```

In the third tutorial, we have incorporated lining and hanging figures by creating two font families: a family with the basic, three-character font family name (lining figures) and a second family featuring hanging figures, with the letter `j` appended to the font family name. The character `j` is the Fontname code for hanging figures. In this tutorial, we need an additional code: the letter `x`, indicating a font featuring expert glyphs. When installing expert sets with the `\latinfamily` macro we use these family names to instruct fontinst that we have an expert set at hand and that we want it to create a font family featuring expert glyphs with lining figures (5) plus a second family featuring expert glyphs with hanging figures (6):

```
5 \latinfamily{mjnx}{}
6 \latinfamily{mjnj}{}
7 \bye
```

Please note that appending `x` and `j` to the font family name works for expert font sets only. The `\latinfamily` macro is not capable of dealing with `sc` & `osf` font sets in the same way. These sets always require a fontinst file using low-level commands such as the one discussed in tutorial III.

v.2 Verbose fontinst file

While the `\latinfamily` macro incorporates the most fundamental features of expert sets, such as optical small caps and additional f-ligatures, it does not exploit all the glyphs found in expert fonts. To use them, you will need to use low-level fontinst commands, at least for parts of the fontinst file. But before we start with our verbose fontinst file, let's first take a look at some encoding issues specific to expert fonts. When dealing with `sc` & `osf` fonts in the third tutorial, we had to rename some glyphs or move them around because in `sc` & `osf` fonts, hanging figures and small caps are found in the standard slots for figures and the lowercase alphabet. With small caps and hanging figures provided by expert fonts the installation is in fact simpler since all glyph names are unique. To understand the difference, we will take a brief look at the glyph names in the respective `afm` files. Compare the names of lowercase glyphs as found in `mjnr8a.afm` to the small caps glyph names in `mjnr8x.afm`:

```
C 97 ; WX 427 ; N a ; B 59 -13 409 426 ;
C 98 ; WX 479 ; N b ; B 18 -13 442 692 ;
C 99 ; WX 427 ; N c ; B 44 -13 403 426 ;
```

```
C 97 ; WX 479 ; N Asmall ; B 19 -4 460 451 ;
C 98 ; WX 438 ; N Bsmall ; B 31 -4 395 434 ;
C 99 ; WX 500 ; N Csmall ; B 37 -12 459 443 ;
```

The situation is similar for lining and hanging ('old style') figures. The following lines are taken from `mjnr8a.afm` and `mjnr8x.afm` respectively:

```
C 48 ; WX 469 ; N zero ; B 37 -12 432 627 ;
C 49 ; WX 469 ; N one ; B 109 -5 356 625 ;
C 50 ; WX 469 ; N two ; B 44 0 397 627 ;

C 48 ; WX 469 ; N zerooldstyle ; B 39 0 431 387 ;
C 49 ; WX 271 ; N oneoldstyle ; B 44 -5 229 405 ;
C 50 ; WX 396 ; N twooldstyle ; B 37 0 356 415 ;
```

In practice, this means that adding expert fonts to the basic font set amounts to little more than adding them to the input file list of `\installfont` in most cases. Still, some additional steps are required. Fortunately, all we need to do in order to make optical small caps and hanging figures readily available is using dedicated encoding vectors provided by fontinst. These encoding vectors reference the glyphs by names corresponding to those found in expert fonts, thus allowing us to pick optical small caps and hanging figures at will. With that in mind, we can get down to business. Our fontinst file begins with a typical header (1–5):

```
1 \input fontinst.sty
2 \substitutesilent{bx}{b}
3 \substitutesilent{sc}{n}
4 \setint{smallcapsscale}{720}
5 \setint{slant}{167}
```

Unfortunately, Monotype Janson provides small caps for the regular weight only. Hence we have to make do with mechanical small caps for the bold series. We set a scale factor of 0.72 for that (4).

```
6 \transformfont{mjnr8r}{\reencodefont{8r}{\fromafm{mjnr8a}}}
7 \transformfont{mjnr8i}{\reencodefont{8r}{\fromafm{mjnr8a}}}
8 \transformfont{mjnb8r}{\reencodefont{8r}{\fromafm{mjnb8a}}}
9 \transformfont{mjnb8i}{\reencodefont{8r}{\fromafm{mjnb8a}}}
10 \transformfont{mjnr8r}{\slantfont{\int{slant}}\reencodefont{8r}{\fromafm{mjnr8a}}}
11 \transformfont{mjnb8r}{\slantfont{\int{slant}}\reencodefont{8r}{\fromafm{mjnb8a}}}
```

We reencode (6–9) and slant (10–11) the basic fonts as usual. Expert fonts do not require any reencoding, but we do need slanted variants of them as well:

```
12 \transformfont{mjnr8x}{\slantfont{\int{slant}}\fromafm{mjnr8x}}
13 \transformfont{mjnb8x}{\slantfont{\int{slant}}\fromafm{mjnb8x}}
```

We will create two font families: `mjnx`, featuring expert glyphs, optical small caps, and lining figures, plus `mjnj` incorporating hanging instead of lining figures. TS1 encoded virtual fonts will be generated for the `mjnx` family only.

```
14 \installfonts
15 \installfamily{T1}{mjnx}{}
16 \installfamily{TS1}{mjnx}{}
17 \installfont{mjnr9e}{mjnr8r,mjnr8x,latin}{t1}{T1}{mjnx}{m}{n}{}

```

As mentioned above, incorporating expert glyphs boils down to adding an additional file to the arguments of the `\installfont` command, in this case the file `mjnr8x.afm`. Note that we use the encoding suffix `9e` instead of `8t` for all T_1 encoded virtual fonts of the `mjnx` family to indicate that they feature expert glyphs. While the code `8t`, as defined by the Fontname scheme, is for T_1 (Cork) encoding, `9e` indicates T_1 plus expert glyphs. Please refer to section 2.4 of the Fontname scheme for a comprehensive list of these codes and the code tables on page 83 of this guide for additional hints.

```
18 \installfont{mjnrc9e}{mjnr8r,mjnr8x,latin}{t1c}{T1}{mjnx}{m}{sc}{}
```

For the small caps font we use the encoding vector `t1c.etx` which will map the small caps in `mjnr8x.afm` to the encoding positions of the lowercase alphabet in our T_1 encoded virtual font. Instead of `latin.mtx` we use the special metric file `latinsc.mtx` in this case. The remaining virtual fonts of the `mjnx` family are built as expected:

```
19 \installfont{mjnro9e}{mjnro8r,mjnro8x,latin}{t1}{T1}{mjnx}{m}{s1}{}
20 \installfont{mjnri9e}{mjnri8r,mjnri8x,latin}{t1}{T1}{mjnx}{m}{it}{}
21 \installfont{mjnb9e}{mjnb8r,mjnb8x,latin}{t1}{T1}{mjnx}{b}{n}{}
22 \installfont{mjnbc9e}{mjnb8r,mjnb8x,latin}{t1c}{T1}{mjnx}{b}{sc}{}
```

Since the bold expert font does not provide small caps, we create mechanical ones. The `t1c.etx` encoding vector will deal with that transparently, but we have to make sure that the regular `latin.mtx` metric file is read here since there are no optical small caps in the raw font.

```
23 \installfont{mjnbo9e}{mjnbo8r,mjnbo8x,latin}{t1}{T1}{mjnx}{b}{s1}{}
24 \installfont{mjnbi9e}{mjnbi8r,mjnbi8x,latin}{t1}{T1}{mjnx}{b}{it}{}
```

That's it for T_1 encoding. Creating TS_1 encoded virtual fonts featuring expert glyphs is pretty straightforward. We simply add the expert fonts to the input file list. Note the encoding suffix of the virtual fonts. We use `9c` instead of `8c` to indicate that the virtual fonts feature expert glyphs:

```
25 \installfont{mjnr9c}{mjnr8r,mjnr8x,textcomp}{ts1}{TS1}{mjnx}{m}{n}{}
26 \installfont{mjnro9c}{mjnro8r,mjnro8x,textcomp}{ts1}{TS1}{mjnx}{m}{s1}{}
27 \installfont{mjnri9c}{mjnri8r,mjnri8x,textcomp}{ts1}{TS1}{mjnx}{m}{it}{}
28 \installfont{mjnb9c}{mjnb8r,mjnb8x,textcomp}{ts1}{TS1}{mjnx}{b}{n}{}
29 \installfont{mjnbo9c}{mjnbo8r,mjnbo8x,textcomp}{ts1}{TS1}{mjnx}{b}{s1}{}
30 \installfont{mjnbi9c}{mjnbi8r,mjnbi8x,textcomp}{ts1}{TS1}{mjnx}{b}{it}{}
31 \endinstallfonts
```

The `mjnx` family including T_1 and TS_1 encoded fonts is now complete. We continue with the `mjnj` family which we want to feature hanging figures by default:

```
32 \installfonts
33 \installfamily{T1}{mjnj}{}
34 \installfont{mjnr9d}{mjnr8r,mjnr8x,latin}{t1j}{T1}{mjnj}{m}{n}{}
```

The encoding code `9d` indicates a T_1 encoded font with expert glyphs and hanging figures. We will use this code for all T_1 encoded virtual fonts of the `mjnj` family. This family is supposed to feature hanging figures in the standard en-

coding positions for figures. We have to keep in mind that the regular encoding vector for T1 encoding (`t1.etx`) references the figures as “zero,” “one,” “two” while the hanging (‘old style’) figures in the expert font (which we want to be available by default) are labeled “zerooldstyle,” “oneoldstyle” and so on. In order to arrange the glyphs according to our wishes, we could read the regular font, clear the figures, read the expert font and rename the ‘old style’ figures. In this case, however, there is a simpler way: we use the special encoding vector `t1j.etx` which is essentially equivalent to `t1.etx` but automatically appends the suffix “oldstyle” to all figures.

```
35 \installfont{mjnr9d}{mjnr8r,mjnr8x,latin}{t1cj}{T1}{mjnj}{m}{sc}{}

```

We have regular optical small caps, so we use the metric file `latinsc.mtx` here. Instead of `t1c.etx` we use the encoding file `t1cj.etx` to make hanging figures the default. The remaining virtual fonts are built like the upright shape (34):

```
36 \installfont{mjnr9d}{mjnr8r,mjnr8x,latin}{t1j}{T1}{mjnj}{m}{s1}{}
37 \installfont{mjnr9d}{mjnr8r,mjnr8x,latin}{t1j}{T1}{mjnj}{m}{it}{}
38 \installfont{mjnb9d}{mjnb8r,mjnb8x,latin}{t1j}{T1}{mjnj}{b}{n}{}
39 \installfont{mjnb9d}{mjnb8r,mjnb8x,latin}{t1cj}{T1}{mjnj}{b}{sc}{}

```

There are no optical small caps in the bold-weight expert fonts. Thus, when generating the bold small caps font, we use the metric file `latin.mtx` and the encoding file `t1cj.etx` to create mechanical small caps.

```
40 \installfont{mjnb9d}{mjnb8r,mjnb8x,latin}{t1j}{T1}{mjnj}{b}{s1}{}
41 \installfont{mjnb9d}{mjnb8r,mjnb8x,latin}{t1j}{T1}{mjnj}{b}{it}{}
42 \endinstallfonts

```

At this point, we have a comprehensive text setup featuring expert f-ligatures, optical small caps as well as a choice of readily available lining and hanging figures. However, there are some glyphs in expert fonts that we have not considered yet.

v.3 Inferior and superior figures

Expert fonts usually provide superior and inferior figures which can be combined with a dedicated fraction slash called ‘solidus’ to typeset arbitrary text fractions like $\frac{1}{2}$ or even $\frac{31}{127}$. Please note that these figures are not suitable for TeX’s math mode but they can be useful in text mode even if there is no need to typeset text fractions. For example, in this guide the footnote marks in the body text are typeset using superior figures and inferior figures are used for the line numbers of the code listings. Like hanging figures, we want inferior and superior figures to be readily available. Therefore, we will create two additional font families, `mjn0` and `mjn1`, which put inferior and superior figures in the standard encoding positions for figures just like our `mjnj` family does for hanging figures. We have been using the encoding vector `t1j.etx` to make hanging figures the default in this tutorial so let’s find out what `t1j.etx` does in detail and try to modify this approach according to our needs. This is `t1j.etx`:

```

\relax
\encoding
\setcommand\lc#1#2{#2}
\setcommand\uc#1#2{#1}
\setcommand\lctop#1#2{#2}
\setcommand\uctop#1#2{#1}
\setcommand\lclig#1#2{#2}
\setcommand\uclig#1#2{#1}
\setcommand\digit#1{#1oldstyle}
\inputetx{T1}
\endencoding

```

As you can see, `t1j.etx` is short. It does not define any encoding slots. All it does is predefine a few macros and use `\inputetx` to load `t1.etx` afterwards. The relevant part (and the only point at which it differs from what `t1.etx` does in this respect) is the line defining the `\digit` macro. To understand this mechanism, we need to take a look at how `t1.etx` defines the encoding slots for all figures:

```

\setslot{\digit{one}}\endsetslot
\setslot{\digit{two}}\endsetslot
\setslot{\digit{three}}\endsetslot

```

The glyph names of figures are not given verbatim, they are used as an argument to the `\digit` macro. The default definition of this macro as given in `t1.etx` looks like this:

```

\setcommand\digit#1{#1}

```

This means that the glyph labeled “one” in the `afm` file will end up in the encoding position for the numeral one in the virtual font – and so on. `t1j.etx` predefines the `\digit` macro as follows:

```

\setcommand\digit#1{#1oldstyle}

```

In this case the glyph labeled “oneoldstyle” in the `afm` file will end up in the encoding position for the numeral one in the `T1` encoded virtual font. When taking a look at the glyph names of hanging, inferior, and superior figures in the `afm` files of our expert fonts now, the approach we need to take in order to access them should be obvious:

```

C 48 ; WX 469 ; N zerooldstyle ; B 39 0 431 387 ;
C 49 ; WX 271 ; N oneoldstyle ; B 44 -5 229 405 ;
C 50 ; WX 396 ; N twooldstyle ; B 37 0 356 415 ;

C 210 ; WX 323 ; N zeroinferior ; B 27 -13 296 355 ;
C 211 ; WX 323 ; N oneinferior ; B 84 -5 240 357 ;
C 212 ; WX 323 ; N twoinferior ; B 27 0 288 358 ;

C 200 ; WX 323 ; N zerosuperior ; B 27 293 296 661 ;
C 201 ; WX 323 ; N onesuperior ; B 84 298 240 661 ;
C 202 ; WX 323 ; N twosuperior ; B 27 303 288 661 ;

```

Just like ‘old style’ figures, inferior and superior figures use suffixes to the respective glyph names in (properly encoded) expert fonts. This means that we

can modify `t1j.etx` accordingly to create encoding vectors incorporating inferior and superior figures. Hence our encoding vector for `T1` encoded fonts featuring inferior figures (`t10.etx`, read: t-one-zero since 0 is the Fontname code for inferior figures) should look like this:

```
\relax
\encoding
\setcommand\lc#1#2{#2}
\setcommand\uc#1#2{#1}
\setcommand\lctop#1#2{#2}
\setcommand\uctop#1#2{#1}
\setcommand\lclig#1#2{#2}
\setcommand\uclig#1#2{#1}
\setcommand\digit#1{#1inferior}
\inputetx{t1}
\endencoding
```

All we need to do in `t10.etx` is use `\setcommand` to predefine the `\digit` macro as follows:

```
\setcommand\digit#1{#1inferior}
```

This will add the suffix “inferior” to all digits. For superior figures, the approach is similar. We create an encoding vector called `t11.etx` (read: t-one-one since 1 is the Fontname code for superior figures):

```
\relax
\encoding
\setcommand\lc#1#2{#2}
\setcommand\uc#1#2{#1}
\setcommand\lctop#1#2{#2}
\setcommand\uctop#1#2{#1}
\setcommand\lclig#1#2{#2}
\setcommand\uclig#1#2{#1}
\setcommand\digit#1{#1superior}
\inputetx{t1}
\endencoding
```

With `t10.etx` and `t11.etx` at hand, we may now create the font families `mjn0` and `mjn1` pretty much like we have generated `mjnj`. Let’s put the new encoding vectors in our working directory and go back to the `fontinst` file:

```
43 \installfonts
44 \installfamily{T1}{mjn0}{}
45 \installfont{mjnr09e}{mjnr8r,mjnr8x,latin}{t10}{T1}{mjn0}{m}{n}{}
```

We add the code 0 to the name of the virtual font (`mjnr09e` here), use the encoding vector `t10.etx`, and adapt the NFSS font declaration (in this case `T1/mjn0/m/n`) accordingly. Other than that, the virtual fonts of the `mjn0` family are generated in the usual way:

```
46 \installfont{mjnr09e}{mjnr8r,mjnr8x,latin}{t10}{T1}{mjn0}{m}{sl}{}
47 \installfont{mjnr09e}{mjnr8r,mjnr8x,latin}{t10}{T1}{mjn0}{m}{it}{}
48 \installfont{mjnb09e}{mjnb8r,mjnb8x,latin}{t10}{T1}{mjn0}{b}{n}{}
49 \installfont{mjnb09e}{mjnb8r,mjnb8x,latin}{t10}{T1}{mjn0}{b}{sl}{}
50 \installfont{mjnb09e}{mjnb8r,mjnb8x,latin}{t10}{T1}{mjn0}{b}{it}{}

```

```
51 \endinstallfonts
```

Our fontinst file will omit the small caps shape to save some disk space. We have included a global shape substitution for the `sc` shape in the header, so `mjn0/sc` will be substituted by `mjn0/n` via a silent substitution in the font definition file. Since the figures of upright and small caps shapes do not differ at all and since we need the `mjn0` family for figures only, we can safely omit the small caps shape. For the `mjn1` family, we adapt the names of the virtual fonts (adding the Fontname code 1 to indicate superior figures), the encoding vector (`t11.etx`), and the `NFSS` declaration in a similar way:

```
52 \installfonts
53 \installfamily{T1}{mjn1}{}
54 \installfont{mjnr19e}{mjnr8r,mjnr8x,latin}{t11}{T1}{mjn1}{m}{n}{}
55 \installfont{mjnro19e}{mjnro8r,mjnro8x,latin}{t11}{T1}{mjn1}{m}{s1}{}
56 \installfont{mjnri19e}{mjnri8r,mjnri8x,latin}{t11}{T1}{mjn1}{m}{it}{}
57 \installfont{mjnb19e}{mjnb8r,mjnb8x,latin}{t11}{T1}{mjn1}{b}{n}{}
58 \installfont{mjnbo19e}{mjnbo8r,mjnbo8x,latin}{t11}{T1}{mjn1}{b}{s1}{}
59 \installfont{mjnbi19e}{mjnbi8r,mjnbi8x,latin}{t11}{T1}{mjn1}{b}{it}{}
60 \endinstallfonts
61 \bye
```

This is our complete fontinst file which will provide us with four font families: `mjnx`, `mjnj`, `mjn0`, and `mjn1`. Virtual fonts in `T1` encoding are provided for all families, but `TS1` encoded ones for `mjnx` only since they would be identical for all of our four font families anyway. Thus, we can simply use substitutions instead of creating duplicate virtual fonts. As mentioned in the third tutorial, however, fontinst does not provide family substitutions. We have to write font definition files manually to ensure that the lacking `TS1` encoded fonts are substituted by their counterparts of the `mjnx` family so that the `textcomp` package will work with all of them. For the `mjnj` family, our font definition file for `TS1` encoding (`ts1mjnj.fd`) looks like this:

```
\ProvidesFile{ts1mjnj.fd}
\DeclareFontFamily{TS1}{mjnj}{}
\DeclareFontShape{TS1}{mjnj}{m}{n}{<-> ssub * mjnx/m/n}{}
\DeclareFontShape{TS1}{mjnj}{m}{sc}{<-> ssub * mjnx/m/n}{}
\DeclareFontShape{TS1}{mjnj}{m}{s1}{<-> ssub * mjnx/m/s1}{}
\DeclareFontShape{TS1}{mjnj}{m}{it}{<-> ssub * mjnx/m/it}{}
\DeclareFontShape{TS1}{mjnj}{b}{n}{<-> ssub * mjnx/b/n}{}
\DeclareFontShape{TS1}{mjnj}{b}{sc}{<-> ssub * mjnx/b/n}{}
\DeclareFontShape{TS1}{mjnj}{b}{s1}{<-> ssub * mjnx/b/s1}{}
\DeclareFontShape{TS1}{mjnj}{b}{it}{<-> ssub * mjnx/b/it}{}
\DeclareFontShape{TS1}{mjnj}{bx}{n}{<-> ssub * mjnx/b/n}{}
\DeclareFontShape{TS1}{mjnj}{bx}{sc}{<-> ssub * mjnx/b/n}{}
\DeclareFontShape{TS1}{mjnj}{bx}{s1}{<-> ssub * mjnx/b/s1}{}
\DeclareFontShape{TS1}{mjnj}{bx}{it}{<-> ssub * mjnx/b/it}{}
\endinput
```

This is the equivalent for `mjn0`, the file `ts1mjn0.fd`:

```
\ProvidesFile{ts1mjn0.fd}
\DeclareFontFamily{TS1}{mjn0}{}
\DeclareFontShape{TS1}{mjn0}{m}{n}{<-> ssub * mjnx/m/n}{}
\endinput
```

```

\DeclareFontShape{TS1}{mjn0}{m}{sc}{<-> ssub * mjnx/m/n} {}
\DeclareFontShape{TS1}{mjn0}{m}{sl}{<-> ssub * mjnx/m/sl} {}
\DeclareFontShape{TS1}{mjn0}{m}{it}{<-> ssub * mjnx/m/it} {}
\DeclareFontShape{TS1}{mjn0}{b}{n}{<-> ssub * mjnx/b/n} {}
\DeclareFontShape{TS1}{mjn0}{b}{sc}{<-> ssub * mjnx/b/n} {}
\DeclareFontShape{TS1}{mjn0}{b}{sl}{<-> ssub * mjnx/b/sl} {}
\DeclareFontShape{TS1}{mjn0}{b}{it}{<-> ssub * mjnx/b/it} {}
\DeclareFontShape{TS1}{mjn0}{bx}{n}{<-> ssub * mjnx/b/n} {}
\DeclareFontShape{TS1}{mjn0}{bx}{sc}{<-> ssub * mjnx/b/n} {}
\DeclareFontShape{TS1}{mjn0}{bx}{sl}{<-> ssub * mjnx/b/sl} {}
\DeclareFontShape{TS1}{mjn0}{bx}{it}{<-> ssub * mjnx/b/it} {}
\endinput

```

And finally, `ts1mjn1.fd` for the `mjn1` family:

```

\ProvidesFile{ts1mjn1.fd}
\DeclareFontFamily{TS1}{mjn1}{}
\DeclareFontShape{TS1}{mjn1}{m}{n}{<-> ssub * mjnx/m/n} {}
\DeclareFontShape{TS1}{mjn1}{m}{sc}{<-> ssub * mjnx/m/n} {}
\DeclareFontShape{TS1}{mjn1}{m}{sl}{<-> ssub * mjnx/m/sl} {}
\DeclareFontShape{TS1}{mjn1}{m}{it}{<-> ssub * mjnx/m/it} {}
\DeclareFontShape{TS1}{mjn1}{b}{n}{<-> ssub * mjnx/b/n} {}
\DeclareFontShape{TS1}{mjn1}{b}{sc}{<-> ssub * mjnx/b/n} {}
\DeclareFontShape{TS1}{mjn1}{b}{sl}{<-> ssub * mjnx/b/sl} {}
\DeclareFontShape{TS1}{mjn1}{b}{it}{<-> ssub * mjnx/b/it} {}
\DeclareFontShape{TS1}{mjn1}{bx}{n}{<-> ssub * mjnx/b/n} {}
\DeclareFontShape{TS1}{mjn1}{bx}{sc}{<-> ssub * mjnx/b/n} {}
\DeclareFontShape{TS1}{mjn1}{bx}{sl}{<-> ssub * mjnx/b/sl} {}
\DeclareFontShape{TS1}{mjn1}{bx}{it}{<-> ssub * mjnx/b/it} {}
\endinput

```

As far as LaTeX is concerned, our setup is complete now. We still need a map file, though.

v.4 The map file

The syntax of map files has been discussed in detail before. The lines for the basic font set should therefore be obvious:

```

mjnr8r JansonMT "TeXBase1Encoding ReEncodeFont" <8r.enc <mjnr8a.pfb
mjnr8i JansonMT-Italic "TeXBase1Encoding ReEncodeFont" <8r.enc <mjnr8a.pfb
mjnb8r JansonMT-Bold "TeXBase1Encoding ReEncodeFont" <8r.enc <mjnb8a.pfb
mjnb8i JansonMT-BoldItalic "TeXBase1Encoding ReEncodeFont" <8r.enc <mjnb8a.pfb
mjnr08r JansonMT "0.167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc <mjnr8a.pfb
mjnb08r JansonMT-Bold "0.167 SlantFont TeXBase1Encoding ReEncodeFont" <8r.enc <mjnb8a.pfb

```

Mapping lines for expert fonts are simpler because there is no need for reencoding and no encoding vector will be included:

```

mjnr8x JansonExpertMT <mjnr8x.pfb
mjnr8i JansonExpertMT-Italic <mjnr8x.pfb
mjnb8x JansonExpertMT-Bold <mjnb8x.pfb
mjnb8i JansonExpertMT-BoldItalic <mjnb8x.pfb

```

We do need slanted expert fonts as well, though:

```

mjnr08x JansonExpertMT "0.167 SlantFont" <mjnr8x.pfb
mjnb08x JansonExpertMT-Bold "0.167 SlantFont" <mjnb8x.pfb

```

This is our complete map file for Monotype Janson, `mjn.map`.

v.5 The style file

Our style file for Janson, `janson.sty`, is based on the one suggested in section III.3. We simply adjust the package name and the names of the font families:

```

1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{janson}[2002/12/30 v1.0 Monotype Janson]
3 \RequirePackage{T1}{fontenc}
4 \RequirePackage{textcomp}
5 \RequirePackage{nfssex}
6 \DeclareOption{oldstyle}{\renewcommand{\rmdefault}{mjnj}}
7 \DeclareOption{lining}{\renewcommand{\rmdefault}{mjnx}}
8 \ExecuteOptions{oldstyle}
9 \ProcessOptions
10 \endinput

```

With an expert font set at hand, however, we have to extend `nfssex.sty` to support expert families:

```

1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{nfssex}[2003/03/14 v1.2 Experimental NFSS Extensions]
3 \newcommand*\exfs@tempa{}
4 \newcommand*\exfs@tempb{}
5 \newcommand*\exfs@try@family[2][{}]{%
6   \let\exfs@tempa\relax
7   \begingroup
8     \fontfamily{#2}\try@load@fontshape%
9     \expandafter\ifx\csname\curr@fontshape\endcsname\relax
10      \edef\exfs@tempa{#1}%
11      \ifx\exfs@tempa\empty
12        \PackageWarning{nfssex}{%
13          Font family '\f@encoding/#2' not available\MessageBreak
14          Ignoring font switch}%
15      \else
16        \PackageInfo{nfssex}{%
17          Font family '\f@encoding/#2' not available\MessageBreak
18          Font family '\f@encoding/#1' tried instead}%
19        \exfs@try@family{#1}%
20      \fi
21    \else
22      \gdef\exfs@tempa{\fontfamily{#2}\selectfont}%
23    \fi
24  \endgroup
25  \exfs@tempa}

```

As soon as expert fonts come into play, the `\lnstyle` macro has to cater for two font families which, depending on the font, may contain lining figures: a basic font family with a three-character code or an expert family with a four-character code ending with the letter `x`. To make sure that `nfssex.sty` will work for fonts like Janson as well as fonts without an expert set, the first thing we need to do is extend our main font switching macro, enabling it to cope with both cases. To do so, we will introduce an optional argument. Essentially, we try to load the font family given by the mandatory argument first (8). If this family is not available, we do not quit with a warning but add a note to the log file (16–18) and try the family given by the optional argument next (19). If loading

the alternative family fails as well, we finally print a warning message (12–14). If the optional argument is not used, the second step will be omitted.

```
26 \def\exfs@get@base#1#2#3#4\@nil{#1#2#3}
27 \DeclareRobustCommand{\lnstyle}{%
28   \not@math@alphabet\lnstyle\relax
29   \exfs@try@family[\expandafter\exfs@get@base\family\@nil]%
30   {\expandafter\exfs@get@base\family\@nil x}}
```

After that, the `\lnstyle` macro needs to be adjusted in order to exploit the optional argument. It will try the expert family with a four-character code first (30) and make `\exfs@try@family` fall back to the basic font family with a three-character code (29) if the former is not available.

```
31 \DeclareRobustCommand{\osstyle}{%
32   \not@math@alphabet\osstyle\relax
33   \exfs@try@family{\expandafter\exfs@get@base\family\@nil j}}
```

The availability of hanging figures is expressed by appending the letter `j` to the font family code for both basic and expert font sets, so `\osstyle` does not need any modification.

```
34 \DeclareRobustCommand{\instyle}{%
35   \not@math@alphabet\instyle\relax
36   \exfs@try@family{\expandafter\exfs@get@base\family\@nil 0}}
37 \DeclareRobustCommand{\sustyle}{%
38   \not@math@alphabet\sustyle\relax
39   \exfs@try@family{\expandafter\exfs@get@base\family\@nil 1}}
```

With inferior and superior figures implemented as two additional font families, `mjn0` and `mjn1`, we add two macros activating these families by adding `0` and `1` to the family name respectively.

```
40 \DeclareTextFontCommand{\textln}{\lnstyle}
41 \DeclareTextFontCommand{\textos}{\osstyle}
42 \DeclareTextFontCommand{\textin}{\instyle}
43 \DeclareTextFontCommand{\textsu}{\sustyle}
44 \endinput
```

We also add two text commands, `\textin` and `\textsu`, which activate these figures locally, similar to `\textit` or `\textbf`.

v.6 Using the fonts

Most features of expert font sets such as additional f-ligatures and optical small caps will be available automatically when selecting the new font families. Using them does not require any additional macros. Lining and hanging figures can be conveniently selected by activating the respective font family, in this case `mjnx` and `mjnj`, or by using the style file `janson.sty` suggested above. Since inferior and superior figures are not used as regular figures, they are treated differently. We will take a look at some possible applications. The inferior and superior figures found in expert fonts were originally intended for typesetting text fractions so let's write a simple macro for that. To typeset a fraction, we combine inferior

and superior figures with the `\textfractionsolidus` macro provided by the `textcomp` package. Accessing the figures implies switching font families locally. Note the additional set of braces which will keep the font change local:

```
\newcommand*{\textfrac}[2]{%
  {\fontfamily{mjn1}\selectfont #1}%
  \textfractionsolidus
  {\fontfamily{mjn0}\selectfont #2}}
```

Writing `\textfrac{1}{2}` in the input file will typeset the fraction $\frac{1}{2}$. When looking at an expert font in a font editor, you will see that expert fonts contain a fixed number of text fractions. Some of them are included in T₁ encoding and supported by the `textcomp` package, but typing rather long commands such as `\textthreequarters` is not exactly convenient. Since there are only nine of them they are not very useful anyway. With a complete set of inferior and superior figures at our disposal, our macro will work for arbitrary fractions like $\frac{3}{7}$ or $\frac{13}{17}$. Instead of using ‘hard-wired’ fonts as shown above, it is even better to use the font switching macros provided by `nfssect.sty` instead since they will dynamically adjust to the active text font:

```
\newcommand*{\textfrac}[2]{%
  \textsu{#1}%
  \textfractionsolidus
  \textin{#2}}
```

What about using superior figures as footnote numbers? To do so, we need to redefine `\@makefnmark`. This is the default definition:

```
\def\@makefnmark{\hbox{\@textsuperscript{\normal font\@thefnmark}}}
```

In order to use optical superior figures instead of mechanical ones, we drop `\@textsuperscript` and switch font families instead:

```
\def\@makefnmark{\hbox{\fontfamily{mjn1}\selectfont\@thefnmark}}
```

We do not need to add additional braces in this case since `\hbox` will keep the font change local. Using our new font switching macros, this may also be accomplished like this:

```
\def\@makefnmark{\hbox{\sustyle\@thefnmark}}
```

Note that, if you want to put a definition of `\@makefnmark` in the preamble of a regular Latex input file (as opposed to a class or a style file), it has to be enclosed in `\makeatletter` and `\makeatother`:

```
\makeatletter
\def\@makefnmark{\hbox{\sustyle\@thefnmark}}
\makeatother
```


TUTORIAL VI

EXPERT FONT SETS, EXTENDED SETUP

In this tutorial we will combine what we have learned in tutorials III and V to install a very complete font set featuring expert fonts, small caps, and hanging figures. This tutorial will also add multiple weights, italic small caps, italic swashes and text ornaments to that. Our example is Adobe Minion, base plus expert packages:

pmnr8a	Minion-Regular	A	143	morg_____
pmnrI8a	Minion-Italic	A	143	moi_____
pmns8a	Minion-Semibold	A	143	mosb_____
pmnsI8a	Minion-SemiboldItalic	A	143	mosbi_____
pmnb8a	Minion-Bold	A	143	mob_____
pmnbI8a	Minion-BoldItalic	A	143	mobi_____
pmnc8a	Minion-Black	A	143	mobl_____
pmnrc8a	Minion-RegularSC	A	144	mosc_____
pmnrIc8a	Minion-ItalicSC	A	144	moisc_____
pmnrIw7a	Minion-SwashItalic	A	144	moswi_____
pmnsc8a	Minion-SemiboldSC	A	144	mosbs_____
pmnsIc8a	Minion-SemiboldItalicSC	A	144	mosic_____
pmnsIw7a	Minion-SwashSemiboldItalic	A	144	mossb_____
pmnbj8a	Minion-BoldOsF	A	144	mobos_____
pmnbjI8a	Minion-BoldItalicOsF	A	144	mobio_____
pmncj8a	Minion-BlackOsF	A	144	mozof_____
pmnr8x	MinionExp-Regular	A	144	mjrg_____
pmnrI8x	MinionExp-Italic	A	144	mji_____
pmns8x	MinionExp-Semibold	A	144	mjsb_____
pmnsI8x	MinionExp-SemiboldItalic	A	144	mjsbi_____
pmnb8x	MinionExp-Bold	A	144	mjb_____
pmnbI8x	MinionExp-BoldItalic	A	144	mjbi_____
pmnc8x	MinionExp-Black	A	144	mjbl_____
pmnrp	Minion-Ornaments	A	144	moor_____

In addition to these fonts, the expert package includes a set of regular-weight display fonts intended for titling and display work at very large sizes. Generated from the same master sources by interpolation, the display fonts share the lettershapes of the text fonts while being based on a design size of 72 pt. Since they form a complete set including small caps and expert fonts, they are handled just like the Minion text set and we will not explicitly consider them here.

VI.1 The fontinst file

With a very comprehensive set of fonts at our disposal, we will be fastidious. There will be no computed glyph shapes – no mechanical small caps and no slanted fonts – thus making this setup suitable for professional typesetting. Note that the bold and black weights do not feature optical small caps. Even though there are expert fonts for these weights, they do not contain any small caps glyphs. The bold weight is merely intended for applications requiring a

very strong contrast, for example to highlight the keywords in a dictionary, while the black weight of a typeface like Minion is only relevant for certain types of display work. Without further ado, we start off as usual:

```
1 \nonstopmode
2 \input fontinst.sty
3 \substitutesilent{bx}{sb}
4 \substitutenoisy{sc}{n}
5 \substitutenoisy{si}{it}
```

When looking at our font set it is obvious that semibold should be used as the main bold weight, hence we make it the default by substituting `sb` for `bx`. Since the bold and black weights do not feature optical small caps, we add appropriate substitutions for the `sc` and `si` (italic small caps) shapes.

```
6 \transformfont{pmnr8r}{\reencodefont{8r}{\fromafm{pmnr8a}}}
7 \transformfont{pmnrc8r}{\reencodefont{8r}{\fromafm{pmnrc8a}}}
8 \transformfont{pmnri8r}{\reencodefont{8r}{\fromafm{pmnri8a}}}
9 \transformfont{pmnric8r}{\reencodefont{8r}{\fromafm{pmnric8a}}}
10 \transformfont{pmns8r}{\reencodefont{8r}{\fromafm{pmns8a}}}
11 \transformfont{pmnsc8r}{\reencodefont{8r}{\fromafm{pmnsc8a}}}
12 \transformfont{pmnsi8r}{\reencodefont{8r}{\fromafm{pmnsi8a}}}
13 \transformfont{pmnsic8r}{\reencodefont{8r}{\fromafm{pmnsic8a}}}
14 \transformfont{pmnb8r}{\reencodefont{8r}{\fromafm{pmnb8a}}}
15 \transformfont{pmnbi8r}{\reencodefont{8r}{\fromafm{pmnbi8a}}}
16 \transformfont{pmnc8r}{\reencodefont{8r}{\fromafm{pmnc8a}}}
```

Reencoding: you know the drill. We reencode all base fonts using Adobe Standard as their native encoding. While the swash fonts are based on Adobe Standard as well, they contain a special set of glyphs and are handled like expert fonts.

```
17 \installfonts
18 \installfamily{T1}{pmnx}{}
19 \installfamily{TS1}{pmnx}{}
20 \installfont{pmnr9e}{pmnr8r,pmnr8x,latin}{t1}{T1}{pmnx}{m}{n}{}
21 \installfont{pmnri9e}{pmnri8r,pmnri8x,latin}{t1}{T1}{pmnx}{m}{it}{}

```

The setup of the upright and italic shapes does not differ from tutorial v.

```
22 \installfont{pmnrc9e}%
23   {kernoff,pmnr8r,pmnr8x,kernon,glyphoff,pmnrc8r,glyphon,reset,sc,latin,sc}%
24   {t1c}{T1}{pmnx}{m}{sc}{}
25 \installfont{pmnric9e}%
26   {kernoff,pmnri8r,pmnri8x,kernon,glyphoff,pmnric8r,glyphon,reset,sc,latin,sc}%
27   {t1c}{T1}{pmnx}{m}{si}{}

```

There is one problem with taking optical small caps from an expert font as demonstrated in tutorial v: there are no kerning pairs between the uppercase alphabet and the small caps replacing the lowercase letters. Without dedicated small caps fonts there is nothing we can do about that. Now that we have both expert and small caps fonts, however, we could take an approach similar to the one outlined in tutorial III, adding the expert font on top of them to get the additional ligatures. We will use a different technique though, which extracts the more comprehensive kerning data from the small caps fonts while taking

the glyphs from the base and the expert fonts only. Apart from being conceptually cleaner, this approach has the additional benefit of not requiring the small caps fonts after the metrics and the virtual fonts have been generated, resulting in slightly smaller PDF and Postscript files if fonts are embedded. The input file list should be more or less self-explanatory: we use `kernoff.mtx` to ignore the kerning data while reading the respective base and expert fonts. Then we add `kernon.mtx` to re-activate the kerning commands and a special metric file called `glyphoff.mtx` to ignore the glyph data. After that, we read the corresponding small caps font and re-activate the glyph commands. Finally, we add `resetsc.mtx` as well as `latinsc.mtx`. Our encoding file is `t1c.etx`.

```

28 \installfont{pmns9e}{pmns8r,pmns8x,latin}{t1}{T1}{pmnx}{sb}{n}{}
29 \installfont{pmnsi9e}{pmnsi8r,pmnsi8x,latin}{t1}{T1}{pmnx}{sb}{it}{}
30 \installfont{pmnsc9e}%
31   {kernoff,pmns8r,pmns8x,kernon,glyphoff,pmnsc8r,glyphon,resetsc,latinsc}%
32   {t1c}{T1}{pmnx}{sb}{sc}{}
33 \installfont{pmnsic9e}%
34   {kernoff,pmnsi8r,pmnsi8x,kernon,glyphoff,pmnsic8r,glyphon,resetsc,latinsc}%
35   {t1c}{T1}{pmnx}{sb}{si}{}

```

We repeat these steps for the semibold weight.

```

36 \installfont{pmnb9e}{pmnb8r,pmnb8x,latin}{t1}{T1}{pmnx}{b}{n}{}
37 \installfont{pmnbi9e}{pmnbi8r,pmnbi8x,latin}{t1}{T1}{pmnx}{b}{it}{}
38 \installfont{pmnc9e}{pmnc8r,pmnc8x,latin}{t1}{T1}{pmnx}{eb}{n}{}

```

The bold and black weights are handled differently because there are no optical small caps. We will simply omit the respective shapes. The black weight will be mapped to the `eb` series of the `NFSS`. After finishing `T1` encoding we continue with `TS1`. Our setup for `TS1` encoding does not differ from tutorial `v` either:

```

39 \installfont{pmnr9c}{pmnr8r,pmnr8x,textcomp}{ts1}{TS1}{pmnx}{m}{n}{}
40 \installfont{pmnri9c}{pmnri8r,pmnri8x,textcomp}{ts1}{TS1}{pmnx}{m}{it}{}
41 \installfont{pmns9c}{pmns8r,pmns8x,textcomp}{ts1}{TS1}{pmnx}{sb}{n}{}
42 \installfont{pmnsi9c}{pmnsi8r,pmnsi8x,textcomp}{ts1}{TS1}{pmnx}{sb}{it}{}
43 \installfont{pmnb9c}{pmnb8r,pmnb8x,textcomp}{ts1}{TS1}{pmnx}{b}{n}{}
44 \installfont{pmnbi9c}{pmnbi8r,pmnbi8x,textcomp}{ts1}{TS1}{pmnx}{b}{it}{}
45 \installfont{pmnc9c}{pmnc8r,pmnc8x,textcomp}{ts1}{TS1}{pmnx}{eb}{n}{}
46 \endinstallfonts

```

The `pmnx` family is now complete. We continue with `pmnj` which will feature hanging figures by default:

```

47 \installfonts
48 \installfamily{T1}{pmnj}{}
49 \installfont{pmnr9d}{pmnr8r,pmnr8x,latin}{t1j}{T1}{pmnj}{m}{n}{}
50 \installfont{pmnri9d}{pmnri8r,pmnri8x,latin}{t1j}{T1}{pmnj}{m}{it}{}

```

To make hanging figures the default throughout the `pmnj` family we employ the encoding file `t1j.etx`. Other than that, the setup of the upright and italic shapes does not differ from `pmnx`.

```

51 \installfont{pmnrc9d}
52   {kernoff,pmnr8r,pmnr8x,kernon,glyphoff,pmnrc8r,glyphon,resetosf,resetsc,latinsc}%
53   {t1cj}{T1}{pmnj}{m}{sc}{}

```

```

54 \installfont{pmnric9d}
55   {kernoff,pmnri8r,pmnri8x,kernon,glyphoff,pmnric8r,glyphon,resetosf,resetsc,latinsc}%
56   {t1cj}{T1}{pmnj}{m}{si}{}

```

For the small caps shape of the `pmnj` family we essentially use the technique introduced above. Since this font family will feature hanging figures we use the encoding file `t1cj.etx` and add the metric file `resetosf.mtx`.

```

57 \installfont{pmns9d}{pmns8r,pmns8x,latin}{t1j}{T1}{pmnj}{sb}{n}{}
58 \installfont{pmnsi9d}{pmnsi8r,pmnsi8x,latin}{t1j}{T1}{pmnj}{sb}{it}{}
59 \installfont{pmnsc9d}
60   {kernoff,pmns8r,pmns8x,kernon,glyphoff,pmnsc8r,glyphon,resetosf,resetsc,latinsc}%
61   {t1cj}{T1}{pmnj}{sb}{sc}{}
62 \installfont{pmnsic9d}
63   {kernoff,pmnsi8r,pmnsi8x,kernon,glyphoff,pmnsic8r,glyphon,resetosf,resetsc,latinsc}%
64   {t1cj}{T1}{pmnj}{sb}{si}{}

```

Again, we repeat these steps for the semibold weight.

```

65 \installfont{pmnb9d}{pmnb8r,pmnb8x,latin}{t1j}{T1}{pmnj}{b}{n}{}
66 \installfont{pmnbi9d}{pmnbi8r,pmnbi8x,latin}{t1j}{T1}{pmnj}{b}{it}{}
67 \installfont{pmnc9d}{pmnc8r,pmnc8x,latin}{t1j}{T1}{pmnj}{eb}{n}{}
68 \endinstallfonts

```

The bold and black weights are essentially handled like those of the `pmnx` family, only differing in the choice of the encoding file.

```

69 \installfonts
70 \installfamily{T1}{pmn0}{}
71 \installfont{pmnr09e}{pmnr8r,pmnr8x,latin}{t10}{T1}{pmn0}{m}{n}{}
72 \installfont{pmnri09e}{pmnri8r,pmnri8x,latin}{t10}{T1}{pmn0}{m}{it}{}
73 \installfont{pmns09e}{pmns8r,pmns8x,latin}{t10}{T1}{pmn0}{sb}{n}{}
74 \installfont{pmnsi09e}{pmnsi8r,pmnsi8x,latin}{t10}{T1}{pmn0}{sb}{it}{}
75 \installfont{pmnb09e}{pmnb8r,pmnb8x,latin}{t10}{T1}{pmn0}{b}{n}{}
76 \installfont{pmnbi09e}{pmnbi8r,pmnbi8x,latin}{t10}{T1}{pmn0}{b}{it}{}
77 \installfont{pmnc09e}{pmnc8r,pmnc8x,latin}{t10}{T1}{pmn0}{eb}{n}{}
78 \endinstallfonts

```

In addition to `pmnx` and `pmnj`, we also add dedicated font families incorporating inferior and superior figures. Since inferior figures are found in the expert fonts, our approach here does not differ from the one introduced in section v.3.

```

79 \installfonts
80 \installfamily{T1}{pmn1}{}
81 \installfont{pmnr19e}{pmnr8r,pmnr8x,latin}{t11}{T1}{pmn1}{m}{n}{}
82 \installfont{pmnri19e}{pmnri8r,pmnri8x,latin}{t11}{T1}{pmn1}{m}{it}{}
83 \installfont{pmns19e}{pmns8r,pmns8x,latin}{t11}{T1}{pmn1}{sb}{n}{}
84 \installfont{pmnsi19e}{pmnsi8r,pmnsi8x,latin}{t11}{T1}{pmn1}{sb}{it}{}
85 \installfont{pmnb19e}{pmnb8r,pmnb8x,latin}{t11}{T1}{pmn1}{b}{n}{}
86 \installfont{pmnbi19e}{pmnbi8r,pmnbi8x,latin}{t11}{T1}{pmn1}{b}{it}{}
87 \installfont{pmnc19e}{pmnc8r,pmnc8x,latin}{t11}{T1}{pmn1}{eb}{n}{}
88 \endinstallfonts

```

The same holds true for superior figures.

```

89 \installfonts
90 \installfamily{T1}{pmnw}{}
91 \installfont{pmnriw9d}{pmnri8r,unsetcaps,pmnriw7a,pmnri8x,latin}{t1j}{T1}{pmnw}{m}{it}{}
92 \installfont{pmnriw9d}{pmnri8r,unsetcaps,pmnriw7a,pmnri8x,latin}{t1j}{T1}{pmnw}{sb}{it}{}

```

```
93 \endinstallfonts
94 \bye
```

In order to incorporate the italic swashes we create an additional font family called `pmnw`. We read the respective base font and clear the slots of the capital letters using the metric file `unsetcaps.mtx`. After that we add the respective swash font and finally the expert font as usual. We employ `t1j.etx` to get hanging figures by default. Our self-made metric file `unsetcaps.mtx` uses the `\unsetglyph` command as follows:

```
\relax
\metrics
\unsetglyph{A}
\unsetglyph{B}
\unsetglyph{C}
...
\unsetglyph{X}
\unsetglyph{Y}
\unsetglyph{Z}
\endmetrics
```

We are merely clearing the slots of capital letters found in the English here. Capital letters with an accent are not removed because the Minion swash set does not provide accented swash capitals anyway. This means that all accented capital letters will be taken from the ordinary italic font. In this particular case `OT1` encoding could be used as a workaround since this encoding constructs accented letters from the English alphabet as discussed in tutorial 1. So here is the respective part of the file for `OT1` encoding:

```
\installfonts
\installfamily{OT1}{pmnw}{}
\installfont{pmnrw9o}{pmnr8r,unsetcaps,pmnrw7a,pmnr8x,latin}{ot1j}{OT1}{pmnw}{m}{it}{}
\installfont{pmnsw9o}{pmnsw8r,unsetcaps,pmnsw7a,pmnsw8x,latin}{ot1j}{OT1}{pmnw}{sb}{it}{}
\endinstallfonts
```

Note that using a setup including `OT1` encoding for one font family only will require switching the encoding explicitly when selecting the swash fonts:

```
\fontencoding{OT1}\fontfamily{pmnw}\selectfont
```

The `pmnw` family as generated by `fontinst` will only cover two shapes in either case. Since `fontinst` does not support family substitutions we cannot take the missing shapes from `pmnj` in the `fontinst` file. We have to edit the respective font definition file, `t1pmnw.fd`, after running `fontinst`. For `T1` encoding it should look as follows:

```
\ProvidesFile{t1pmnw.fd}
\DeclareFontFamily{T1}{pmnw}{}
\DeclareFontShape{T1}{pmnw}{m}{n}{<-> ssub * pmnj/m/n} {}
\DeclareFontShape{T1}{pmnw}{m}{sc}{<-> ssub * pmnj/m/sc} {}
\DeclareFontShape{T1}{pmnw}{m}{sl}{<-> ssub * pmnj/m/sl} {}
\DeclareFontShape{T1}{pmnw}{m}{it}{<-> pmnrw9d} {}
\DeclareFontShape{T1}{pmnw}{m}{si}{<-> ssub * pmnj/m/si} {}
\DeclareFontShape{T1}{pmnw}{sb}{n}{<-> ssub * pmnj/sb/n} {}
```

```

\DeclareFontShape{T1}{pmnw}{sb}{sc}{<-> ssub * pmnj/sb/sc}{}
\DeclareFontShape{T1}{pmnw}{sb}{sl}{<-> ssub * pmnj/sb/sl}{}
\DeclareFontShape{T1}{pmnw}{sb}{it}{<-> pmnsw9d}{}
\DeclareFontShape{T1}{pmnw}{sb}{si}{<-> ssub * pmnj/sb/si}{}
\DeclareFontShape{T1}{pmnw}{b}{n}{<-> ssub * pmnj/b/n}{}
\DeclareFontShape{T1}{pmnw}{b}{sc}{<-> ssub * pmnj/b/sc}{}
\DeclareFontShape{T1}{pmnw}{b}{sl}{<-> ssub * pmnj/b/sl}{}
\DeclareFontShape{T1}{pmnw}{b}{it}{<-> ssub * pmnj/b/it}{}
\DeclareFontShape{T1}{pmnw}{b}{si}{<-> ssub * pmnj/b/si}{}
\DeclareFontShape{T1}{pmnw}{eb}{n}{<-> ssub * pmnj/eb/n}{}
\DeclareFontShape{T1}{pmnw}{eb}{sc}{<-> ssub * pmnj/eb/sc}{}
\DeclareFontShape{T1}{pmnw}{eb}{sl}{<-> ssub * pmnj/eb/sl}{}
\DeclareFontShape{T1}{pmnw}{eb}{it}{<-> ssub * pmnj/eb/it}{}
\DeclareFontShape{T1}{pmnw}{eb}{si}{<-> ssub * pmnj/eb/si}{}
\DeclareFontShape{T1}{pmnw}{bx}{n}{<-> ssub * pmnw/bx/n}{}
\DeclareFontShape{T1}{pmnw}{bx}{sc}{<-> ssub * pmnw/bx/sc}{}
\DeclareFontShape{T1}{pmnw}{bx}{sl}{<-> ssub * pmnw/bx/sl}{}
\DeclareFontShape{T1}{pmnw}{bx}{it}{<-> ssub * pmnw/bx/it}{}
\DeclareFontShape{T1}{pmnw}{bx}{si}{<-> ssub * pmnw/bx/si}{}
\endinput

```

Only the pmnx family offers TS1 encoded fonts as the glyphs found in this encoding are identical across all font families. To make sure that all font families work as expected, however, we need font definition files containing family substitutions which cannot be defined in a fontinst file. For the pmnj family:

```

\ProvidesFile{ts1pmnj.fd}
\DeclareFontFamily{TS1}{pmnj}{}
\DeclareFontShape{TS1}{pmnj}{m}{n}{<-> ssub * pmnx/m/n}{}
\DeclareFontShape{TS1}{pmnj}{m}{sc}{<-> ssub * pmnx/m/sc}{}
\DeclareFontShape{TS1}{pmnj}{m}{sl}{<-> ssub * pmnx/m/sl}{}
\DeclareFontShape{TS1}{pmnj}{m}{it}{<-> ssub * pmnx/m/it}{}
\DeclareFontShape{TS1}{pmnj}{sb}{n}{<-> ssub * pmnx/sb/n}{}
\DeclareFontShape{TS1}{pmnj}{sb}{sc}{<-> ssub * pmnx/sb/sc}{}
\DeclareFontShape{TS1}{pmnj}{sb}{sl}{<-> ssub * pmnx/sb/sl}{}
\DeclareFontShape{TS1}{pmnj}{sb}{it}{<-> ssub * pmnx/sb/it}{}
\DeclareFontShape{TS1}{pmnj}{b}{n}{<-> ssub * pmnx/b/n}{}
\DeclareFontShape{TS1}{pmnj}{b}{sc}{<-> ssub * pmnx/b/sc}{}
\DeclareFontShape{TS1}{pmnj}{b}{sl}{<-> ssub * pmnx/b/sl}{}
\DeclareFontShape{TS1}{pmnj}{b}{it}{<-> ssub * pmnx/b/it}{}
\DeclareFontShape{TS1}{pmnj}{eb}{n}{<-> ssub * pmnx/eb/n}{}
\DeclareFontShape{TS1}{pmnj}{eb}{sc}{<-> ssub * pmnx/eb/sc}{}
\DeclareFontShape{TS1}{pmnj}{eb}{sl}{<-> ssub * pmnx/eb/sl}{}
\DeclareFontShape{TS1}{pmnj}{eb}{it}{<-> ssub * pmnx/eb/it}{}
\DeclareFontShape{TS1}{pmnj}{bx}{n}{<-> ssub * pmnx/bx/n}{}
\DeclareFontShape{TS1}{pmnj}{bx}{sc}{<-> ssub * pmnx/bx/sc}{}
\DeclareFontShape{TS1}{pmnj}{bx}{sl}{<-> ssub * pmnx/bx/sl}{}
\DeclareFontShape{TS1}{pmnj}{bx}{it}{<-> ssub * pmnx/bx/it}{}
\endinput

```

For the pmnw family:

```

\ProvidesFile{ts1pmnw.fd}
\DeclareFontFamily{TS1}{pmnw}{}
\DeclareFontShape{TS1}{pmnw}{m}{n}{<-> ssub * pmnx/m/n}{}
\DeclareFontShape{TS1}{pmnw}{m}{sc}{<-> ssub * pmnx/m/sc}{}
\DeclareFontShape{TS1}{pmnw}{m}{sl}{<-> ssub * pmnx/m/sl}{}
\DeclareFontShape{TS1}{pmnw}{m}{it}{<-> ssub * pmnx/m/it}{}

```

```

\DeclareFontShape{TS1}{pmnw}{sb}{n}{<->ssub*pmnx/sb/n}{}
\DeclareFontShape{TS1}{pmnw}{sb}{sc}{<->ssub*pmnx/sb/n}{}
\DeclareFontShape{TS1}{pmnw}{sb}{sl}{<->ssub*pmnx/sb/sl}{}
\DeclareFontShape{TS1}{pmnw}{sb}{it}{<->ssub*pmnx/sb/it}{}
\DeclareFontShape{TS1}{pmnw}{b}{n}{<->ssub*pmnx/b/n}{}
\DeclareFontShape{TS1}{pmnw}{b}{sc}{<->ssub*pmnx/b/n}{}
\DeclareFontShape{TS1}{pmnw}{b}{sl}{<->ssub*pmnx/b/sl}{}
\DeclareFontShape{TS1}{pmnw}{b}{it}{<->ssub*pmnx/b/it}{}
\DeclareFontShape{TS1}{pmnw}{eb}{n}{<->ssub*pmnx/eb/n}{}
\DeclareFontShape{TS1}{pmnw}{eb}{sc}{<->ssub*pmnx/eb/n}{}
\DeclareFontShape{TS1}{pmnw}{eb}{sl}{<->ssub*pmnx/eb/sl}{}
\DeclareFontShape{TS1}{pmnw}{eb}{it}{<->ssub*pmnx/eb/it}{}
\DeclareFontShape{TS1}{pmnw}{bx}{n}{<->ssub*pmnx/sb/n}{}
\DeclareFontShape{TS1}{pmnw}{bx}{sc}{<->ssub*pmnx/sb/n}{}
\DeclareFontShape{TS1}{pmnw}{bx}{sl}{<->ssub*pmnx/sb/sl}{}
\DeclareFontShape{TS1}{pmnw}{bx}{it}{<->ssub*pmnx/sb/it}{}
\endinput

```

For the pmn0 family:

```

\ProvidesFile{ts1pmn0.fd}
\DeclareFontFamily{TS1}{pmn0}{}
\DeclareFontShape{TS1}{pmn0}{m}{n}{<->ssub*pmnx/m/n}{}
\DeclareFontShape{TS1}{pmn0}{m}{sc}{<->ssub*pmnx/m/n}{}
\DeclareFontShape{TS1}{pmn0}{m}{sl}{<->ssub*pmnx/m/sl}{}
\DeclareFontShape{TS1}{pmn0}{m}{it}{<->ssub*pmnx/m/it}{}
\DeclareFontShape{TS1}{pmn0}{sb}{n}{<->ssub*pmnx/sb/n}{}
\DeclareFontShape{TS1}{pmn0}{sb}{sc}{<->ssub*pmnx/sb/n}{}
\DeclareFontShape{TS1}{pmn0}{sb}{sl}{<->ssub*pmnx/sb/sl}{}
\DeclareFontShape{TS1}{pmn0}{sb}{it}{<->ssub*pmnx/sb/it}{}
\DeclareFontShape{TS1}{pmn0}{b}{n}{<->ssub*pmnx/b/n}{}
\DeclareFontShape{TS1}{pmn0}{b}{sc}{<->ssub*pmnx/b/n}{}
\DeclareFontShape{TS1}{pmn0}{b}{sl}{<->ssub*pmnx/b/sl}{}
\DeclareFontShape{TS1}{pmn0}{b}{it}{<->ssub*pmnx/b/it}{}
\DeclareFontShape{TS1}{pmn0}{eb}{n}{<->ssub*pmnx/eb/n}{}
\DeclareFontShape{TS1}{pmn0}{eb}{sc}{<->ssub*pmnx/eb/n}{}
\DeclareFontShape{TS1}{pmn0}{eb}{sl}{<->ssub*pmnx/eb/sl}{}
\DeclareFontShape{TS1}{pmn0}{eb}{it}{<->ssub*pmnx/eb/it}{}
\DeclareFontShape{TS1}{pmn0}{bx}{n}{<->ssub*pmnx/sb/n}{}
\DeclareFontShape{TS1}{pmn0}{bx}{sc}{<->ssub*pmnx/sb/n}{}
\DeclareFontShape{TS1}{pmn0}{bx}{sl}{<->ssub*pmnx/sb/sl}{}
\DeclareFontShape{TS1}{pmn0}{bx}{it}{<->ssub*pmnx/sb/it}{}
\endinput

```

For the pmn1 family:

```

\ProvidesFile{ts1pmn1.fd}
\DeclareFontFamily{TS1}{pmn1}{}
\DeclareFontShape{TS1}{pmn1}{m}{n}{<->ssub*pmnx/m/n}{}
\DeclareFontShape{TS1}{pmn1}{m}{sc}{<->ssub*pmnx/m/n}{}
\DeclareFontShape{TS1}{pmn1}{m}{sl}{<->ssub*pmnx/m/sl}{}
\DeclareFontShape{TS1}{pmn1}{m}{it}{<->ssub*pmnx/m/it}{}
\DeclareFontShape{TS1}{pmn1}{sb}{n}{<->ssub*pmnx/sb/n}{}
\DeclareFontShape{TS1}{pmn1}{sb}{sc}{<->ssub*pmnx/sb/n}{}
\DeclareFontShape{TS1}{pmn1}{sb}{sl}{<->ssub*pmnx/sb/sl}{}
\DeclareFontShape{TS1}{pmn1}{sb}{it}{<->ssub*pmnx/sb/it}{}
\DeclareFontShape{TS1}{pmn1}{b}{n}{<->ssub*pmnx/b/n}{}
\DeclareFontShape{TS1}{pmn1}{b}{sc}{<->ssub*pmnx/b/n}{}
\DeclareFontShape{TS1}{pmn1}{b}{sl}{<->ssub*pmnx/b/sl}{}

```

```

\DeclareFontShape{TS1}{pmn1}{b}{it}{<-> ssub * pmnx/b/it} {}
\DeclareFontShape{TS1}{pmn1}{eb}{n} {<-> ssub * pmnx/eb/n} {}
\DeclareFontShape{TS1}{pmn1}{eb}{sc}{<-> ssub * pmnx/eb/n} {}
\DeclareFontShape{TS1}{pmn1}{eb}{sl}{<-> ssub * pmnx/eb/sl} {}
\DeclareFontShape{TS1}{pmn1}{eb}{it}{<-> ssub * pmnx/eb/it} {}
\DeclareFontShape{TS1}{pmn1}{bx}{n} {<-> ssub * pmnx/sb/n} {}
\DeclareFontShape{TS1}{pmn1}{bx}{sc}{<-> ssub * pmnx/sb/n} {}
\DeclareFontShape{TS1}{pmn1}{bx}{sl}{<-> ssub * pmnx/sb/sl} {}
\DeclareFontShape{TS1}{pmn1}{bx}{it}{<-> ssub * pmnx/sb/it} {}
\endinput

```

VI.2 Text ornaments

The Minion expert package includes a dedicated ornament font, `pmnrrp.pfb`. As discussed before in section IV.3, we do not really need `fontinst` when installing symbol fonts. Since no reencoding is required and there are no virtual fonts, `afm2tfm` is sufficient for the job:

```
afm2tfm pmnrrp.afm pmnrrp.tfm
```

Using the fonts with LaTeX requires a font definition file, though. Symbol fonts are not based on any particular encoding, so we use the encoding code U (uncoded, unknown) in this case. This is `upmnp.fd`:

```

\ProvidesFile{upmnp.fd}
\DeclareFontFamily{U}{pmnp}{}
\DeclareFontShape{U}{pmnp}{m}{n}{<-> pmnrrp} {}
\endinput

```

VI.3 The map file

The map file for Minion is longer than the one in the last tutorial, but conceptually similar. Note that the `sc` & `osf` fonts are not required. They are included here for reference only. The swash and ornament fonts were not reencoded, hence their mapping is similar to that of expert fonts:

```

pmnr8r Minion-Regular "TeXBase1Encoding ReEncodeFont" <8r.enc <pmnr8a.pfb
pmnr18r Minion-Italic "TeXBase1Encoding ReEncodeFont" <8r.enc <pmnr18a.pfb
pmns8r Minion-Semibold "TeXBase1Encoding ReEncodeFont" <8r.enc <pmns8a.pfb
pmns18r Minion-SemiboldItalic "TeXBase1Encoding ReEncodeFont" <8r.enc <pmns18a.pfb
pmnb8r Minion-Bold "TeXBase1Encoding ReEncodeFont" <8r.enc <pmnb8a.pfb
pmnb18r Minion-BoldItalic "TeXBase1Encoding ReEncodeFont" <8r.enc <pmnb18a.pfb
pmnc8r Minion-Black "TeXBase1Encoding ReEncodeFont" <8r.enc <pmnc8a.pfb
pmnrc8r Minion-RegularSC "TeXBase1Encoding ReEncodeFont" <8r.enc <pmnrc8a.pfb
pmnric8r Minion-ItalicSC "TeXBase1Encoding ReEncodeFont" <8r.enc <pmnric8a.pfb
pmnsc8r Minion-SemiboldSC "TeXBase1Encoding ReEncodeFont" <8r.enc <pmnsc8a.pfb
pmnsic8r Minion-SemiboldItalicSC "TeXBase1Encoding ReEncodeFont" <8r.enc <pmnsic8a.pfb
pmnbj8r Minion-Bold0sF "TeXBase1Encoding ReEncodeFont" <8r.enc <pmnbj8a.pfb
pmnbj18r Minion-BoldItalic0sF "TeXBase1Encoding ReEncodeFont" <8r.enc <pmnbj18a.pfb
pmncj8r Minion-Black0sF "TeXBase1Encoding ReEncodeFont" <8r.enc <pmncj8a.pfb
pmnr8x MinionExp-Regular <pmnr8x.pfb
pmnr18x MinionExp-Italic <pmnr18x.pfb
pmns8x MinionExp-Semibold <pmns8x.pfb
pmns18x MinionExp-SemiboldItalic <pmns18x.pfb
pmnb8x MinionExp-Bold <pmnb8x.pfb

```


pmnbi8x	MinionExp-BoldItalic	<pmnbi8x.pfb
pmnc8x	MinionExp-Black	<pmnc8x.pfb
pmnrw7a	Minion-SwashItalic	<pmnrw7a.pfb
pmnsiw7a	Minion-SwashSemiboldItalic	<pmnsiw7a.pfb
pmnrrp	Minion-Ornaments	<pmnrrp.pfb

VI.4 Extending the user interface

Before creating a style file for Minion, we will update `nfssect.sty` one more time to support its additional features. Support for swashes is easily added since the framework is already in place. Therefore, the first part of the file does not require any changes, we simply add support for swashes by defining `\swstyle` in a similar vein (40–42):

```

1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{nfssect}[2003/03/14 v1.2 Experimental NFSS Extensions]
3 \newcommand*\exfs@tempa{}{}
4 \newcommand*\exfs@tempb{}{}
5 \newcommand*\exfs@try@family[2][{}]{%
6   \let\exfs@tempa\relax
7   \begingroup
8     \fontfamily{#2}\try@load@fontshape
9     \expandafter\ifx\csname\curr@fontshape\endcsname\relax
10    \edef\exfs@tempa{#1}%
11    \ifx\exfs@tempa\empty
12      \PackageWarning{nfssect}{%
13        Font family '\f@encoding/#2' not available\MessageBreak
14        Ignoring font switch}%
15    \else
16      \PackageInfo{nfssect}{%
17        Font family '\f@encoding/#2' not available\MessageBreak
18        Font family '\f@encoding/#1' tried instead}%
19      \exfs@try@family{#1}%
20    \fi
21  \else
22    \gdef\exfs@tempa{\fontfamily{#2}\selectfont}%
23  \fi
24 \endgroup
25 \exfs@tempa}
26 \def\exfs@get@base#1#2#3#4\@nil{#1#2#3}
27 \DeclareRobustCommand{\lnstyle}{%
28   \not@math@alphabet\lnstyle\relax
29   \exfs@try@family[\expandafter\exfs@get@base\f@family\@nil]{%
30     {\expandafter\exfs@get@base\f@family\@nil x}}
31 \DeclareRobustCommand{\osstyle}{%
32   \not@math@alphabet\osstyle\relax
33   \exfs@try@family{\expandafter\exfs@get@base\f@family\@nil j}}
34 \DeclareRobustCommand{\instyle}{%
35   \not@math@alphabet\instyle\relax
36   \exfs@try@family{\expandafter\exfs@get@base\f@family\@nil 0}}
37 \DeclareRobustCommand{\sustyle}{%
38   \not@math@alphabet\sustyle\relax
39   \exfs@try@family{\expandafter\exfs@get@base\f@family\@nil 1}}
40 \DeclareRobustCommand{\swstyle}{%
41   \not@math@alphabet\swstyle\relax
42   \exfs@try@family{\expandafter\exfs@get@base\f@family\@nil w}}

```

Adding thorough support for italic small caps is not quite as easy. The problem is that the creators of the NFSS apparently did not think of italic small caps when putting italics and small caps in the same category. Since both variants are on the shape axis of the NFSS they are mutually exclusive. While this will not keep us from using `\fontshape` to select italic small caps explicitly, nesting `\scshape` and `\itshape` does not have the desired effect. When nested, these macros simply override each other instead of switching to italic small caps. This problem is not as exotic as it may seem because italic small caps are hardly ever used explicitly. Typically, they come into play when small caps and italics are mixed on the same line. For example, think of a page header which is set in small caps, containing a highlighted word set in italics; or an italic section heading with an acronym set in small caps. To work around this problem, we will have to redefine a few NFSS macros. But first of all, we will add a macro for explicit switching to italic small caps.

```
43 \newcommand*{\sdefault}{si}
```

Note that the NFSS does not use fixed shape codes like `it` and `sc` for the italic and the small caps shape, but rather macros like `\itdefault` and `\scdefault`. We will handle italic small caps in a similar way by defining `\sdefault`, which defaults to `si`. Now let's define `\sishape` for explicit switching to italic small caps:

```
44 \DeclareRobustCommand{\sishape}{%
45   \not@math@alphabet\sishape\relax
46   \fontshape\sdefault\selectfont}
```

While we are able to typeset italic small caps by selecting them explicitly, macros like `\itshape` and `\scshape` will simply ignore the new shape. Let's redefine these macros to make them take advantage of italic small caps transparently. In order to do so, we need a macro that will merge properties of the shape axis, thereby allowing us to treat italics and small caps as if they were not on the same axis:

```
47 \newcommand*{\exfs@merge@shape}[3]{%
48   \edef\exfs@tempa{#1}%
49   \edef\exfs@tempb{#2}%
50   \ifx\f@shape\exfs@tempb
51     \expandafter\ifx\csname\f@encoding/\f@family/\f@series/#3\endcsname\relax
52   \else
53     \edef\exfs@tempa{#3}%
54     \fi
55   \fi
56   \fontshape{\exfs@tempa}\selectfont}
```

This macro will switch to the font shape given as the first argument unless the current shape is identical to the one indicated by the second argument. In this case it will switch to the shape designated by the third argument instead, provided that it is available for the current font family. With this macro at hand we redefine `\itshape`:

```

57 \DeclareRobustCommand{\itshape}{%
58   \not@math@alphabet\itshape\mathit
59   \exfs@merge@shape{\itdefault}{\scdefault}{\sdefault}}

```

Essentially, `\itshape` will switch to the font shape `it` unless the current shape is `sc`, in which case it will switch to `si` instead, provided that `si` is available. `\scshape` does it the other way around:

```

60 \DeclareRobustCommand{\scshape}{%
61   \not@math@alphabet\scshape\relax
62   \exfs@merge@shape{\scdefault}{\itdefault}{\sdefault}}

```

We also redefine `\upshape` to make it switch to `sc` instead of `n` if the current shape is `si`:

```

63 \DeclareRobustCommand{\upshape}{%
64   \not@math@alphabet\upshape\relax
65   \exfs@merge@shape{\updefault}{\sdefault}{\scdefault}}

```

If no italic small caps are available, all of these macros will behave like they did before, making them suitable for global use. While we are at it, we also define a new macro, `\dfshape`, that will reset the current shape to the default (`n` unless `\shapedefault` has been redefined) regardless of the current shape:

```

66 \DeclareRobustCommand{\dfshape}{%
67   \not@math@alphabet\dfshape\relax
68   \fontshape\shapedefault\selectfont}

```

Before we add text commands for our new font switches, there is still one thing left to do. The macro `\swstyle`, which we have defined above (40–42), will switch to the the font family providing italic swashes (for example, `pmnw`). However, it will not activate the italic shape. It would be convenient to have a macro which takes care of all of that. We first create an auxiliary macro holding the shape which provides the actual swashes:

```

69 \newcommand*{\swshapedefault}{\itdefault}

```

Then we create a macro which will call `\swstyle` and select the shape providing the italic swashes in one shot:

```

70 \DeclareRobustCommand{\swshape}{%
71   \not@math@alphabet\swshape\relax
72   \swstyle\fontshape\swshapedefault\selectfont}

```

Finally, we add text commands for our new font switches:

```

73 \DeclareTextFontCommand{\textln}{\lnstyle}
74 \DeclareTextFontCommand{\textos}{\osstyle}
75 \DeclareTextFontCommand{\textin}{\instyle}
76 \DeclareTextFontCommand{\textsu}{\sustyle}
77 \DeclareTextFontCommand{\textsi}{\sishape}
78 \DeclareTextFontCommand{\textdf}{\dfshape}
79 \DeclareTextFontCommand{\textsw}{\swshape}

```

As far as text is concerned, all features of Minion are readily available at this point. Using the ornaments would still require low-level commands, though.

VI.5 A high-level interface for ornaments

Technically, ornament fonts are comparable to the euro fonts discussed in section IV.3. To typeset the first ornament of Minion, for example, we could use the following construct:

```
{\usefont{U}{pmpn}{m}{n}\char 97}
```

As this is rather awkward and requires looking at the `afm` file to find out the encoding slot of each ornament, we will implement a higher-level solution. The problem is that ornament fonts do not conform to any encoding, so there is no standard we could rely on as far as the order of the glyphs in the font is concerned. We have to provide this information explicitly in `minion.sty`. To facilitate this, we define the following macro:

```
80 \newcommand*{\DeclareTextOrnament}[7]{%
81   \expandafter\def\csname#1@orn\@roman#2\endcsname{#3/#4/#5/#6/#7}}
```

To declare the first ornament of Minion, this macro would be employed as follows:

```
\DeclareTextOrnament{pmn}{1}{U}{pmpn}{m}{n}{97}
```

We use the first three letters of the font family name as an identifier (`pmn`) and assign a number (1 in this case) to the ornament defined by the remaining arguments. These arguments form a complete font declaration with a syntax similar to that of the `NFSS` macro `\DeclareFontShape`. The last argument is the encoding slot of the ornament (97 here) as given in the `afm` file. You might wonder why we use a complete font declaration here. Since all ornaments are located in the same font, using the same encoding, series, and shape, this seems to be redundant. In this case, this is actually true. The problem is that ornaments are not necessarily provided in dedicated fonts. Adobe Garamond, for example, comes with ornaments which are included in some of the alternate text fonts so we use a complete declaration for maximum flexibility. Internally, the ornaments are saved in a format modeled after the way the `NFSS` handles font shapes. When typesetting an ornament later, we need a macro to parse this font declaration:

```
82 \begingroup
83   \catcode'\=12
84   \gdef\exfs@split@orndef#1/#2/#3/#4/#5\@nil{%
85     \def\f@encoding{#1}%
86     \def\f@family{#2}%
87     \def\f@series{#3}%
88     \def\f@shape{#4}%
89     \def\exfs@tempa{#5}}
90 \endgroup
```

Since we use the base of the font family name as an identifier, we also need a macro that expands to the first three letters of the current font family:

```
91 \def\exfs@base@family{\expandafter\exfs@get@base\f@family\@nil}
```

Now we can finally implement a user macro that actually typesets the ornament. We will simply call it `\ornament`:

```

92 \DeclareRobustCommand{\ornament}[1]{%
93   \expandafter\ifx\csname\exfs@base@family @orn\@roman#1\endcsname\relax
94     \PackageWarning{nfssect}{%
95       Ornament #1 undefined for font family '\exfs@base@family'\MessageBreak
96       Setting debug mark}%
97     \rule{1ex}{1ex}%
98   \else
99     \begingroup
100    \edef\exfs@tempb{\csname\exfs@base@family @orn\@roman#1\endcsname}%
101    \expandafter\expandafter\expandafter\exfs@split@orndef
102    \expandafter\string\exfs@tempb\@nil
103    \selectfont\char\exfs@tempa
104    \endgroup
105   \fi}
106 \endinput

```

First of all, we check if the desired ornament has been declared (93) and issue a warning if not (94–96). We also typeset a mark (97) to facilitate debugging in this case. If it has been declared, we expand and parse the declaration (100–102), switch fonts, and typeset the ornament (103). We use a group to keep the font change local.

vi.6 The style file

The style file for Minion is similar to the ones suggested in section III.3 and v.5. The only difference is the declaration of the text ornaments. This is the first part of `minion.sty`:

```

1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{minion}[2003/03/25 v1.0 Adobe Minion]
3 \RequirePackage[T1]{fontenc}
4 \RequirePackage{textcomp}
5 \RequirePackage{nfssect}
6 \DeclareOption{oldstyle}{\renewcommand*{\rmdefault}{\pmmj}}
7 \DeclareOption{lining}{\renewcommand*{\rmdefault}{\pmtx}}
8 \ExecuteOptions{oldstyle}
9 \ProcessOptions

```

When declaring the text ornaments, we take the encoding slot numbers from the respective afm file:

```

C 97 ; WX 885 ; N ornament1 ; B 50 -65 835 744 ;
C 98 ; WX 1036 ; N ornament2 ; B 50 4 986 672 ;
C 99 ; WX 1066 ; N ornament3 ; B 50 -106 1016 745 ;
C 100 ; WX 866 ; N ornament4 ; B 50 98 816 534 ;
C 101 ; WX 390 ; N ornament5 ; B 50 86 341 550 ;

```

We add a declaration for each ornament:

```

10 \DeclareTextOrnament{pmm}{1}{U}{pmp}{m}{n}{97}
11 \DeclareTextOrnament{pmm}{2}{U}{pmp}{m}{n}{98}
12 \DeclareTextOrnament{pmm}{3}{U}{pmp}{m}{n}{99}
13 \DeclareTextOrnament{pmm}{4}{U}{pmp}{m}{n}{100}
14 \DeclareTextOrnament{pmm}{5}{U}{pmp}{m}{n}{101}

```

```

15 \DeclareTextOrnament{pmn}{6}{U}{pmnp}{m}{n}{102}
16 \DeclareTextOrnament{pmn}{7}{U}{pmnp}{m}{n}{103}
17 \DeclareTextOrnament{pmn}{8}{U}{pmnp}{m}{n}{104}
18 \DeclareTextOrnament{pmn}{9}{U}{pmnp}{m}{n}{105}
19 \DeclareTextOrnament{pmn}{10}{U}{pmnp}{m}{n}{106}
20 \DeclareTextOrnament{pmn}{11}{U}{pmnp}{m}{n}{107}
21 \DeclareTextOrnament{pmn}{12}{U}{pmnp}{m}{n}{108}
22 \DeclareTextOrnament{pmn}{13}{U}{pmnp}{m}{n}{109}
23 \DeclareTextOrnament{pmn}{14}{U}{pmnp}{m}{n}{110}
24 \DeclareTextOrnament{pmn}{15}{U}{pmnp}{m}{n}{111}
25 \DeclareTextOrnament{pmn}{16}{U}{pmnp}{m}{n}{112}
26 \DeclareTextOrnament{pmn}{17}{U}{pmnp}{m}{n}{113}
27 \DeclareTextOrnament{pmn}{18}{U}{pmnp}{m}{n}{114}
28 \DeclareTextOrnament{pmn}{19}{U}{pmnp}{m}{n}{115}
29 \DeclareTextOrnament{pmn}{20}{U}{pmnp}{m}{n}{116}
30 \DeclareTextOrnament{pmn}{21}{U}{pmnp}{m}{n}{117}
31 \DeclareTextOrnament{pmn}{22}{U}{pmnp}{m}{n}{118}
32 \DeclareTextOrnament{pmn}{23}{U}{pmnp}{m}{n}{119}
33 \endinput

```

As mentioned before, Adobe Garamond features ornaments in the alternate text fonts, requiring a complete font declaration. In this case, the definitions would look as follows:

```

\DeclareTextOrnament{pad}{1}{U}{pada}{m}{n}{49}
\DeclareTextOrnament{pad}{2}{U}{pada}{m}{n}{50}
\DeclareTextOrnament{pad}{3}{U}{pada}{m}{it}{49}

```

Note that the ornament macro is deliberately designed to be sensitive to the active font family. When using Minion as text font, `\ornament{1}` will typeset the symbol \mathfrak{E}^{\flat} . When using Adobe Garamond, the same command sequence will typeset \mathfrak{E}^{\flat} instead. If you would like to use these text ornaments in a font independent manner, simply switch font families explicitly, adding extra braces to keep the font change local:

```
{\fontfamily{pmnx}\selectfont\ornament{1}}
```

Which Minion font family you select (for example, `pmnx` or `pmnj`) does not matter, but it has to be a known one, that is, there has to be a font definition file corresponding to the active text encoding in addition to the one for the ornament font. Note that, since the ornament declarations are given in the style file, you also need to load the respective package in the document preamble. For example, if you would like to typeset a document in Sabon and make use of some Minion text ornaments, you might do the following:

```

\documentclass...
\usepackage{minion}
\usepackage{sabon}
...
\begin{document}
...
Text in Sabon
...
{\fontfamily{pmnx}\selectfont\ornament{1}}

```

Apart from that, you can always go back to lower-level commands which merely depend on a font definition file (`upmnp.fd` and `upada.fd` here) for the respective ornament font:

```
{\usefont{U}{pmpn}{m}{n}\char 97}  
{\usefont{U}{pada}{m}{n}\char 49}
```


CODE TABLES

The tables on the following pages are intended to give an idea of how the codes of the Fontname scheme relate to those used by Latex’s font selection scheme (NFSS). The Fontname codes are what we use when renaming the font files during the installation while the NFSS codes are what we need when selecting a certain font under Latex later. Sticking to the NFSS codes listed below is not a technical requirement for a functional font installation. When using the `\latinfamily` macro, `fontinst` will indeed use these NFSS codes. When employing low-level `fontinst` commands, however, the NFSS font declaration is controlled by the last five arguments of the `\installfont` command. In theory, we could use an arbitrary code and the NFSS would handle that just fine. It is still highly recommended to stick to these codes to avoid confusion and incompatibility. Two dashes in one of the table cells indicate that there is no customary code for this font property in the respective scheme whereas a blank cell means that the code is omitted. Properties which are not catered for by the `\latinfamily` macro are marked with an asterisk in the last column.

Please note that Fontname codes and NFSS codes cannot be mapped on a one-to-one basis in all cases since the two schemes are rather different in concept. Weights and widths, which are treated separately by the Fontname scheme, need to be concatenated and handled as a ‘series’ when using the NFSS since the latter does not have independent categories (‘axes’) for weight and width. The ‘variant’ category of the Fontname scheme on the other hand, which embraces several different properties including shapes like italics as well as special glyph sets such as small caps or alternative figures, does not correspond to a single NFSS axis. Some variants, like italics and small caps for example, are mapped to the ‘shape’ axis of the NFSS. Others, such as alternative figures, are handled in completely different ways. Table 5 lists variants corresponding to the most common NFSS shapes only. When looking at the documentation of the Fontname scheme, you will find a lot more variant codes not mentioned here. Although they are used for file naming, they do not, or, at least do not necessarily correspond to a customary NFSS shape. Hanging, inferior, and superior numbers (Fontname codes `j`, `0`, and `1`), for example, are treated as ‘variants’ by the Fontname scheme but they are usually implemented as independent font families on the level of the NFSS. For the encodings listed in table 6 the situation is similar. For example, a virtual font in T1 encoding featuring expert glyphs is indicated by adding `9e` to the file name. However, on the level of the NFSS the encoding code is T1 for all T1 encoded fonts and the fact that the font provides expert glyphs is expressed by adding the letter `x` to the font family name.

WEIGHT	FONTNAME CODE	NFSS SERIES
ultra light, thin, hairline	a	ul [*]
extra light	j	el [*]
light	l	l
book	k	m
regular	r	m
medium	m	mb
demibold	d	db
semibold	s	sb
bold	b	b
heavy	h	eb
black	c	eb
extra bold, extra black	x	eb
ultra bold, ultra black	u	ub
poster	p	-- [*]

TABLE 3: Codes for font weights

WIDTH	FONTNAME CODE	NFSS SERIES
ultra compressed	u	uc [*]
ultra condensed	o	uc [*]
extra compressed, extra condensed	q	ec [*]
compressed	p	c [*]
condensed	c	c [*]
narrow	n	c
regular		
extended	x	x [*]
expanded	e	x [*]
extra expanded	v	ex [*]
ultra expanded	--	ux [*]
wide	w	-- [*]

TABLE 4: Codes for font widths

VARIANT	FONTNAME CODE	NFSS SHAPE
normal, upright, roman		n
italic	i	it
oblique, slanted	o	sl
small caps	c	sc
italic small caps	ic	si [*]
upright italic	--	ui [*]
outline	l	ol [*]

TABLE 5: Codes for font variants

ENCODING	FONTNAME CODE	NFSS ENCODING
Adobe Standard	8a	8a
Expert	8x	8x
Tex Base 1	8r	8r
Tex Text	7t	OT1
Tex Tex with expert set	9t	OT1
Tex Text with expert set and osf	9o	OT1
Cork	8t	T1
Cork with expert set	9e	T1
Cork with expert set and osf	9d	T1
Text Companion	8c	TS1
Text Companion with expert set	9c	TS1

TABLE 6: Codes for font encodings

THE GNU FREE DOCUMENTATION LICENSE

Version 1.2, November 2002

Copyright © 2000, 2001, 2002 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

o. Preamble

The purpose of this license is to make a manual, textbook, or other functional and useful document ‘free’ in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this license preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This license is a kind of ‘copyleft’, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this license in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this license is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this license principally for works whose purpose is instruction or reference.

1. Applicability and definitions

This license applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this license. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The *document*, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as *you*. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A *modified version* of the document means any work containing the document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A *secondary section* is a named appendix or a front-matter section of the document that deals exclusively with the relationship of the publishers or authors of the document to the document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the document is in part a textbook of mathematics, a secondary section may

not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The *invariant sections* are certain secondary sections whose titles are designated, as being those of invariant sections, in the notice that says that the document is released under this license. If a section does not fit the above definition of secondary then it is not allowed to be designated as invariant. The document may contain zero invariant sections. If the document does not identify any invariant sections then there are none.

The *cover texts* are certain short passages of text that are listed, as front-cover texts or back-cover texts, in the notice that says that the document is released under this license. A front-cover text may be at most five words, and a back-cover text may be at most 25 words.

A *transparent* copy of the document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not transparent. An image format is not transparent if used for any substantial amount of text. A copy that is not ‘transparent’ is called ‘opaque’.

Examples of suitable formats for transparent copies include plain Ascii without markup, Texinfo input format, Latex input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, Postscript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, Postscript or PDF produced by some word processors for output purposes only.

The *title page* means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this license requires to appear in the title page. For works in formats which do not have any title page as such, ‘title page’ means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section *entitled xyz* means a named subunit of the document whose title either is precisely xyz or contains xyz in parentheses following text that translates xyz in another language. (Here xyz stands for a specific section name mentioned below, such as ‘Acknowledgements’, ‘Dedications’, ‘Endorsements’, or ‘History’.) To ‘preserve the title’ of such a section when you modify the docu-

ment means that it remains a section ‘entitled xyz’ according to this definition.

The document may include warranty disclaimers next to the notice which states that this license applies to the document. These warranty disclaimers are considered to be included by reference in this license, but only as regards disclaiming warranties: any other implication that these warranty disclaimers may have is void and has no effect on the meaning of this license.

2. Verbatim copying

You may copy and distribute the document in any medium, either commercially or noncommercially, provided that this license, the copyright notices, and the license notice saying this license applies to the document are reproduced in all copies, and that you add no other conditions whatsoever to those of this license. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. Copying in quantity

If you publish printed copies (or copies in media that commonly have printed covers) of the document, numbering more than 100, and the document’s license notice requires cover texts, you must enclose the copies in covers that carry, clearly and legibly, all these cover texts: front-cover texts on the front cover, and back-cover texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute opaque copies of the document numbering more than 100, you must either include a machine-readable transparent copy along with each opaque copy, or state in or with each opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete transparent copy of the document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of opaque copies in quantity, to ensure that this transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an opaque copy (directly or through your agents or retailers) of that edition to

the public.

It is requested, but not required, that you contact the authors of the document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the document.

4. Modifications

You may copy and distribute a modified version of the document under the conditions of sections 2 and 3 above, provided that you release the modified version under precisely this license, with the modified version filling the role of the document, thus licensing distribution and modification of the modified version to whoever possesses a copy of it. In addition, you must do these things in the modified version:

- A. Use in the title page (and on the covers, if any) a title distinct from that of the document, and from those of previous versions (which should, if there were any, be listed in the history section of the document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the title page, as authors, one or more persons or entities responsible for authorship of the modifications in the modified version, together with at least five of the principal authors of the document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the title page the name of the publisher of the modified version, as the publisher.
- D. Preserve all the copyright notices of the document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the modified version under the terms of this license.
- G. Preserve in that license notice the full lists of invariant sections and required cover texts given in the document's license notice.
- H. Include an unaltered copy of this license.
- I. Preserve the section entitled 'History', preserve its title, and add to it an item stating at least the title, year, new authors, and publisher of the modified version as given on the title page. If there is no section entitled 'History' in the document, create one stating the title, year, authors, and publisher of the document as given on its title page, then add an item describing the modified version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the document for public access to a transparent copy of the document, and likewise the network locations given in the document for previous versions it was based on. These

may be placed in the ‘History’ section. You may omit a network location for a work that was published at least four years before the document itself, or if the original publisher of the version it refers to gives permission.

- k. For any section entitled ‘Acknowledgements’ or ‘Dedications’, preserve the title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- l. Preserve all the invariant sections of the document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- m. Delete any section entitled ‘Endorsements’. Such a section may not be included in the modified version.
- n. Do not retitle any existing section to be entitled ‘Endorsements’ or to conflict in title with any invariant section.
- o. Preserve any warranty disclaimers.

If the modified version includes new front-matter sections or appendices that qualify as secondary sections and contain no material copied from the document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of invariant sections in the modified version’s license notice. These titles must be distinct from any other section titles.

You may add a section entitled ‘Endorsements’, provided it contains nothing but endorsements of your modified version by various parties – for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a front-cover text, and a passage of up to 25 words as a back-cover text, to the end of the list of cover texts in the modified version. Only one passage of front-cover text and one of back-cover text may be added by (or through arrangements made by) any one entity. If the document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the document do not by this license give permission to use their names for publicity for or to assert or imply endorsement of any modified version.

5. Combining documents

You may combine the document with other documents released under this license, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the invariant sections of all of the original documents, unmodified, and list them all as invariant sections of

your combined work in its license notice, and that you preserve all their warranty disclaimers.

The combined work need only contain one copy of this license, and multiple identical invariant sections may be replaced with a single copy. If there are multiple invariant sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of invariant sections in the license notice of the combined work.

In the combination, you must combine any sections entitled ‘History’ in the various original documents, forming one section entitled ‘History’; likewise combine any sections entitled ‘Acknowledgements’, and any sections entitled ‘Dedications’. You must delete all sections entitled ‘Endorsements’.

6. Collections of documents

You may make a collection consisting of the document and other documents released under this license, and replace the individual copies of this license in the various documents with a single copy that is included in the collection, provided that you follow the rules of this license for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this license, provided you insert a copy of this license into the extracted document, and follow this license in all other respects regarding verbatim copying of that document.

7. Aggregation with independent works

A compilation of the document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an ‘aggregate’ if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the document is included in an aggregate, this license does not apply to the other works in the aggregate which are not themselves derivative works of the document.

If the cover text requirement of section 3 is applicable to these copies of the document, then if the document is less than one half of the entire aggregate, the document’s cover texts may be placed on covers that bracket the document within the aggregate, or the electronic equivalent of covers if the document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. Translation

Translation is considered a kind of modification, so you may distribute translations of the document under the terms of section 4. Replacing invariant sections

with translations requires special permission from their copyright holders, but you may include translations of some or all invariant sections in addition to the original versions of these invariant sections. You may include a translation of this license, and all the license notices in the document, and any warranty disclaimers, provided that you also include the original English version of this license and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this license or a notice or disclaimer, the original version will prevail.

If a section in the document is entitled ‘Acknowledgements’, ‘Dedications’, or ‘History’, the requirement (section 4) to preserve its title (section 1) will typically require changing the actual title.

9. Termination

You may not copy, modify, sublicense, or distribute the document except as expressly provided for under this license. Any other attempt to copy, modify, sublicense or distribute the document is void, and will automatically terminate your rights under this license. However, parties who have received copies, or rights, from you under this license will not have their licenses terminated so long as such parties remain in full compliance.

10. Future revisions of this license

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.¹

Each version of the license is given a distinguishing version number. If the document specifies that a particular numbered version of this license “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the document does not specify a version number of this license, you may choose any version ever published (not as a draft) by the Free Software Foundation.

¹ <http://www.gnu.org/copyleft/>

REVISION HISTORY

- 1.23 2003-08-31 Added tables 1 and 2 to section 1.6
Revised section 1.6, adding note about Latin Modern
Revised section 1.5
- 1.20 2003-07-17 Added preliminary hints concerning fontinst 1.9 to section IV.1
Fixed problem with `nfsssect.sty` (sections III.3, V.5, and VI.4)
Improved support for swashes in `nfsssect.sty` (section VI.4)
Revised discussion of ornaments in section VI.6
Revised discussion of swashes in section VI.1
Added spelling corrections by William Adams
Added spelling corrections by Adrian Burd
- 1.10 2003-03-27 Added GNU Free Documentation License to the appendix
Added explicit licensing clause
- 1.00 2003-03-25 Final release for fontinst 1.8
Added tutorial VI
Updated notes on contributions
- 0.80 2003-03-23 Added spelling corrections and suggestions by Timothy Eyre
Revised section 1.5, splitting off section 1.6
Added section III.4
Updated notes on contributions
- 0.68 2003-02-09 Revised section IV.2
Updated notes on contributions
- 0.66 2003-01-26 Added highlighting to code listings
- 0.65 2003-01-19 Added spelling corrections by Adrian Heathcote
Added spelling corrections by William Adams
Added section II.2
Revised section II.3
Revised introduction
- 0.60 2003-01-11 Revised tutorial III
Added discussion of kerning issues to section III.1
- 0.54 2003-01-04 Revised discussion of OT1 encoding in tutorial I
Added minor changes to code tables
- 0.52 2003-01-02 Added code table 4 to appendix
Revised note on code tables in appendix
- 0.50 2002-12-30 Added tutorial V
Added code tables 3, 5, and 6 to appendix
Added revision history
- 0.43 2002-10-25 First public pre-release featuring tutorials I–IV
Added installation instructions to section IV.3
Added section IV.4
- 0.40 2002-08-11 Added tutorial IV
- 0.30 2002-05-12 Added tutorial III
- 0.20 2002-04-17 Unreleased draft including tutorials I and II